

Amazing Computing™

Premiere Issue
Vol. 1 / Number 1
U.S.A. \$2.50
Canada \$3.50

Commodore Amiga™ Information Programs

Inside CLI

CLI Command Summary

Tutorials in
'C' and Lisp

Program Listings:

SuperSpheres

EZ-Term

Terminal Program

AMICUS™ Section:

AMICUS News

Amiga Public Domain
Software



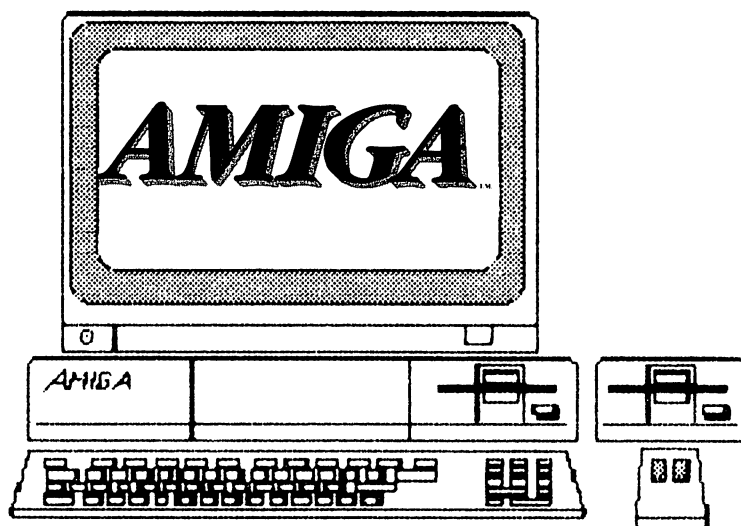
Plus much more:

Feed your AMIGA™ Right

The User's Issue

Amazing Computing™

AMIGA™ Information and Programs



Feed your Amiga RIGHT! Don't miss an issue!!!

Charter Subscriptions

12 Informative Issues

\$24 United States

\$30 Canada and Mexico

\$35 Overseas

Send check or money order to:

PiM PUBLICATIONS, Inc.

P.O. Box 869, Fall River, MA. 02722

Amiga™ is a trademark of Commodore-Amiga, Inc.

Amazing Dealers

The following are **Amazing Dealers**, dedicated to supporting as well as selling the Commodore-Amiga™. They carry **Amazing Computing™**, your resource for information on the Amiga™.

If you are not an **Amazing Dealer**, but would like to become one, contact:

PiM Publications
P.O. Box 869
Fall River, MA. 02722
1-617-679-3109

United States

Alabama

Madison Books & Computers
 Mel's Photo & Computer Shop

Madison
 Montgomery

Arizona

Computer West

Phoenix

Arkansas

The Micro Shop
 SIS Inc.

Little Rock
 Ft. Home

California

Computer Attic
 Brown Knows Computing
 Computer Nook, Inc.
 Home Computing Center
 ComputerLand Santa Rosa
 ComputerLand of Stockton
 HT Electronics

Palo Alto
 Redlands
 San Bernardino
 San Bruno
 Santa Rosa
 Stockton
 Sunnyvale

Colorado

Computer Room

Aurora

Connecticut

Connecting Point
 Spectrum Computers

Georgetown
 New Haven

Florida

Micro's Etc.
 MSI Business System
 The Open Door
 Random Access Computers Inc.
 Softwalls Computer Ctr. Inc.
 Computer Image
 Commodore Computer Shop
 Gulf Coast Computer Elect.
 Computer Bar
 Family Computers
 New Age Electronics

Altamonte Springs
 Brooksville
 Cocoa
 Ft. Walton Beach
 Holly Hill
 Miami
 Ocala
 Panama City
 Pensacola
 Sarasota
 St. Petersburg

Georgia

Future System
 Software South Inc.

Augusta
 Savannah

Illinois

Computer Resolution
 Illini Microcomputers Inc.

Champaign
 Naperville

Indiana

Greg Chaney
 Vons Computers

Greenfield
 West Lafayette

Louisiana

The Computer Clinic
 Software Center International

Lafayette
 Metairie

Maryland

The Logical Choice Inc.
 Computer Crafters

Baltimore
 Wheaton

Massachusetts

HCS Computer Center
 Tycom Inc.
 OmniTech Computers

Marshfield
 Pittsfield
 Tewksbury

Michigan

Galaxy Computers
 Slip Disk
 Programs Unlimited
 Slip Disk
 Midland Computer Shop
 Edmunton Computer Center
 Roseville Computer Store
 Pro-Video

Allen Park
 Clarkston
 Grand Rapids
 Madison Heights
 Midland
 Pontiac
 Roseville
 Ukenos

Minnesota

Computers, etc.
 Computer Specialties Plus

Eagan
 Monticello

Missouri

Brands Mark Computers

Kansas City

Nebraska

BULLS-EYE
 Double E Electronics

Lincoln
 Omaha

Nevada

ComputerLand of Carson

Carson City

New Jersey

Family Computer Centers

South Orange

New Mexico

New Horizon Computer System
 Academy Computers

Alamogordo
 Albuquerque

New York

Ray Supply Inc.
 World Computers Inc.
 Computer Outlet
 Software Supermarket
 Castle Computer
 Byte Shop
 Software & Such

Glen Falls
 Hicksville
 Jamestown
 KENMORE
 Latham
 Merrick
 Scotia

North Carolina

Digitz

Raleigh

North Dakota

Computer Associates

Fargo

Ohio

Software Centre International
 North Coast Programming
 Quality Computer Appliances
 Computers Plus of Ohio

North Olmsted
 Solon
 Toledo
 Upper Sandusky

Oklahoma

Colonial Video & Computer
 Video-Comp Inc.

Bartlesville
 Lawton

Oregon

Coackamas Computers
 IB Computers

Coackamas
 Portland

Pennsylvania

Basic Computer Systems

Hermitage

Rhode Island

CompuTopia
 CompuTopia

Providence
 Warwick

Tennessee

Software First

Nashville

Texas

Software Centre
 Micro Search
 The Computer Experience
 Computer Concepts

Dallas
 Huston
 San Antonio
 Waco

Virginia

Virginia Micro Systems

Woodbridge

Wisconsin

Colortron Computers
 TIMW Software Inc.

Racine
 Wausau

Canada

Ontario

Hastings Data Systems
 Arkon Electronics

Kingston
 Toronto

Saskatchewan

Zycon Computing Ltd.

Regina

Amazing Computing

Publisher: Joyce A. Hicks

Circulation Manager: Doris Gamble

Assistant to The Publisher: Robert James Hicks

Corporate Advisor: Robert H. Gamble

Managing Editor: Don Hicks

Hardware Editor: Ernest P. Viveiros

AMICUS Editor: John Foust

Founding Authors:

Kelly Kauffman

John Foust

George Musser Jr.

Perry Kivolowitz

Daniel Zigmond

Bela Lubkin

Don Hicks

The Amigo

Special Thanks to:

Rev. Richard E. Degagne

Rev. Mr. Thomas A. Frechette

Robert H. Bergwall

RESCO, Inc.

John Bellm (CompUtopia)

Ron Lamoureux (CompUtopia)

Dick Hubert

Advertising Sales:

1-617-679-3109

Amazing Computing™ is published by PIM PUBLICATIONS, Inc., P.O. Box 869, Fall River, MA. 02722. Subscriptions: in the U.S., 12 issues for \$24.00; Canada and Mexico, \$30.00; Overseas, \$35.00. Printed in the U.S.A. Copyright © 1986 by PIM Publications, Inc. All rights reserved.

Amazing Computing™

Index of Advertisers

Advanced Systems Design Group	35
Akron Systems Development	47
Cardinal Software	20
Discovery Software	C III
Elcom Software	35
Kurta Corporation	C IV
Lattice, Inc.	36
Micro-Systems Software, Inc.	16
PiM PUBLICATIONS, Inc.	C II, 6,43
UBZ Software	48

Amazing Computing

Table of Contents

SuperSheres by Kelly Kauffman An ABasic graphics program	5
Date Virus by John Foust There may be a disease spreading through your Amiga disks. The "symptoms" and "cure" are discussed	7
EZ-TERM by Kelly Kauffman A 300 Baud ABasic™ terminal program, let your AMIGA communicate!	11
ROOMERS by The Amigo A fast look at the gossip spreading in the Amiga community	18
Miga-Mania by Perry Kivolowitz Perry talks of mouse care, programming fixes and more!	21
INSIDE CLI by George Musser Jr. Guided by insight instead of documentation, George leads us into the workings of AmigaDos™	25
CLI Summary by George Musser Jr. A removable section with the most used CLI commands listed and explained	29
The Amazing C Tutorial by John Foust John begins part one of a multi-part tutorial on the language C.	37
AmigaForum by Bela Lubkin Take a quick trip through the newest Compuserve Forum designed for Amiga users	44
Commodore Amiga Development Program by Don Hicks Want to be a Commodore Developer? We discuss the questions you should ask and where to send the answers	49
Amiga Products Assorted products are listed with prices and delivery dates. A wish list for your Amiga™	50
The Amazing Lisp, A Tutorial by Daniel Zigmond Dan introduces lisp on the Amiga and begins a session of thought to utilize this language for Amiga	52
AMICUS Network by John Foust John brings AMICUS™ to Amazing Computing™ and discusses the excitement of our Amiga network	56
Public Domain Software by John Foust Public Domain software is introduced.	59

From the Editor:

Welcome to the first issue of Amazing Computing™. If you are wondering what this magazine is about, Amazing Computing™ says it all. We are on a quest to experience the Amazing things Computing can do and the Amazing computer to experience, is the Commodore Amiga™.

The Commodore Amiga has received excellent press and the Amiga has earned every line. It is a fantastic machine, not only for its technology and sophistication, but also, for its potential. The Amiga is an excellent "Skeleton Key". It has been designed to grow in any or as many directions as you require or can invent.

To some, this is the same hype they have already heard. Let me make you a promise. We will do everything in our power not to hype the Amiga or its assorted support products. It will be difficult, we know the machine is great, however, we feel that the information we provide will SHOW a product's merit without our bombarding you with pretty words. Our aim is to research the Amiga as **Amiga users** and explore its possibilities: its full potential.

Yes, we are users. Every writer in this magazine has an Amiga and wants to change the world "just a little" with its power. The Amiga's tools and possibilities excite us.

We want to learn about the Amiga, its software, its hardware, and how these products can be implemented in our homes and offices to make us a bit more productive and life a bit more Amazing.

Each issue will have material for the technical user, but we will also be gearing reviews and tutorials for the **New User**. The New User is someone who has no current ambition to program a computer. They want to purchase software and hardware to do the job for which they bought the computer.

With these users in mind, we will review software and hardware by functionality, as well as merit and technology. We will show how a product can be used in your home and business, possibly creating a solution for you or opening an idea of your own.

Amazing Computing™ is an open forum. We want to hear from our readers. Write us with your concerns, tell us of your victories, and dazzle us with your insights.

We are looking to the readership of Amazing Computing™ to provide our articles. If you have a good idea, drop us a line. We are flexible and we do work with starting writers (why not, we are a starting magazine).

Amazing Computing is dedicated to one more thing: Fun. We will play the arcade games, strategy games and whatever else comes along and give a fair appraisal of its value. But, most of all, we will see how much fun we can have. let us all never forget, the Amiga makes a fantastic toy, let's play.



Don Hicks

Managing Editor

SuperSpheres

by
Kelly Kauffman

SuperSpheres was taken from an idea originally penned on an 8-bit Atari, then translated to the Amiga by David Milligan.

In the original version of Spheres, the spheres are created through complex use of sine, cosine, and degrees in trigonometry. The reason for this is due to the absence of a "Circle" command. However, on the Amiga, we DO have the luxury of the "Circle" command.

This latest version of the program is heavily dependent on the Circle command. It is much quicker and faster than the aforementioned method and is more accurate...to a point. After you type in and run this program, you will notice, especially on the smaller "planets," that it draws outside of the planet and puts little "Mountains" on the top and bottom of your planet.

From as near as I can tell, this is a bug in the Circle command rather than in the formulas fed to the Circle command.

You will notice that the program runs considerably faster, with one exception. The part of the program that erases away an area in order to draw a new sphere on it is slower than the original version of Spheres. The time lost here though is amply made up for in the reduced time it takes to actually draw the Sphere.

To exit the program while it is running, press and HOLD the left mouse button until it exits. The program will only exit just before it is ready to erase the area to draw the new Sphere.

As always, have fun, and Enjoy!!!

SuperSphere PROGRAM

by
Kelly Kauffman [CIS#70206,640]

(Note SuperShere is available on Amicus PDS#2 disk)

1000 ' SuperSphere Version 1.0 - - - - 12/06/85

1100 ' By Kelly Kauffman

1200 '

1300 ' This program is derived from the original Atari 8-bit version,
1400 ' and the Amiga translation. The program runs considerably faster
1500 ' due to the extensive use of the "Circle" command, rather than
1600 ' putting points in arrays and drawing lines connecting them. The
1700 ' circles are also much rounder.

1800 '

2400 ' Please leave my name on this, as I have worked VERY HARD in
2500 ' getting the formulas to work correctly....Thanx!!!

2600 '

2700 ' Hold down the left mouse button, while the program is running
2800 ' to exit. Make sure you hold it down---it may take a second.

3000 '

3500 randomize - 1

3600 screen 0,4

```

3700 rgb 0,0,0:rgb 15,0,0
3800 rgb 1,10,10,10
3900 rgb 3,15,6,0:rgb 9,0,0,15
4000 rgb 10,3,6,15:rgb 11,7,7,15
4100 rgb 12,12,0,14:rgb 13,15,2,14
4200 gosub 8700:gosub 8500
4300 scncir
4400 peno 2
4500 a=2
4600 gosub 6500
4700 str=(hi/100)*5:a=int(rnd*14):
    for i=1 to 19
4800 a=a+1:if a>14 then a=2
4900 peno a
5000 hi=hi-str
5100 circle (x,y),wide,hi/wide
5200 next i
5300 return
5400 goto 6400
5500 str=(hi/100)*5
5600 a=int(rnd*14)
5700 for i=1 to 19
5800 a=a+1:if a>14 then a=2
5900 peno a
6000 hi=hi-str
6100 circle (x,y),hi,wide/hi
6200 next i
6300 return
6400 rem end
6500 rem
6600 wide=rnd*100
6700 x=rnd*385
6800 y=rnd*190
6900 hi=wide
7000 ask mouse x%,y%,b%:if b%=4 then 10400
7100 gosub 7600
7200 gosub 4700
7300 hi=wide
7400 gosub 5500
7500 goto 6500
7600 hig=hi
7700 hig=hi
7800 for i=1 to 100
7900 peno 0
8000 circle(x,y),wide,hig/wide
8100 hig=hig-1:if hig<1 then i=100
8200 next i
8300 pena 0:draw (x-wide,y to x+wide,y)
8400 return
8500 window 1,0,0,320,200,
    "SuperSphere V1.0 by Kelly Kauffman"
8600 cmd 1:return

```

```

8700 hi=180
8800 box (0,0;319,199),1
8900 x=151:y=90
9000 wide=180
9100 for i=1 to 100 step .5
9200 circle (x,y),wide,hi/wide
9300 a=a+1:if a>14 then a=2
9400 peno a
9500 pena a-1:outline 1:paint (x,y),1
9600 hi=hi-7
9700 if hi<1 then i=100
9800 next i
9900 drawmode 1:peno 1:pena 4:penb 2:graphic (1)
10000 ? at(100,80);inverse(1);" SuperSphere ";;
    penb 0:pena 9:?at(85,100);inverse(0);
    "by Kelly Kauffman"
10100 ask mouse x%,y%,b%
10200 if b%=0 then 10100
10300 return
10400 close
10500 screen 0,4,0
10600 end

```

•AC•



Amazing Computing

AMIGA™ Information and Programs

Charter Subscriptions

12 Informative Issues

\$24 United States

\$30 Canada and Mexico

\$35 Overseas

PiM PUBLICATIONS, Inc.

P.O. Box 869, Fall River, MA. 02722

1(617)-679-3109

Amiga is a trademark of Commodore-Amiga, Inc.

DATE Virus!

by
John Foust

A friend of mine complained that his Amiga was running slowly. In particular, the **CLI** detailed-directory command, called **'LIST'**, was taking nearly ten seconds to print each line. That seemed impossible. On my machine, **LIST** printed several lines per second.

He later noticed that the system date was set to **'28-Oct-A6'**. He tried to set it back, but it seemed that the date would not change. Many files on this disk carried the strange **'28-Oct-A6'** date.

My friend's disks were infected with the **'date virus'**, an insidious disease that can spread throughout your disk collection in a matter of minutes. The symptoms include dreadfully slow **LIST** and **DATE** commands, file dates of **'Future'**, and the inability to reset the Amiga's internal date.

First, an Amiga anatomy lesson. **AmigaDOS™** is constantly watching the disk drives, ticking each time it checks for the presence of a disk. When a disk is inserted, **AmigaDOS™** tests the integrity of the disk by reading all the directories, looking for bad areas on the disk.

This validation process also examines the disk's **'volume last modified date stamp'**, and each file's date stamp. **Date stamps** have two parts: a day, such as February 12, 1986, and a time, such as 13:54. If any of these dates is more recent than the current internal date and time, it sets the internal date and time ahead to that time.

When you turn on your Amiga, and before you insert the **Kickstart**, the clock is set to January 1, 1978. The operating system considers this day the beginning of time. When you insert the **Workbench**, the disk is validated, and the date is set to the most recent date on the disk.

This system insures that any files you create during this session will have a date stamp more recent than the last = time you used your Amiga. The system date will always advance just past the last time you used your Amiga, if you don't use **Preferences** to set the date, or use the **DATE** command in **CLI**.

But if the system time is set into the future by the **'date virus'**, any files you create on this disk will have the corrupt date stamp, thereby spreading this **'virus'** to other disks. You can contaminate all your disks in a matter of minutes...

Whenever my friend inserted a disk with a bad date, the system time was set to around the year 65,000. At first, only one disk had this bad date. If he inserted another disk, and created a file there, it poisoned that disk, giving it a futuristic date stamp.

The symptom of extremely slow **LIST** commands comes from the way **AmigaDOS™** represents the current year, month and day. The date is stored as the number of days since January 1, 1978. The **DATE** and **LIST** commands also calculate the day of the week, it seems.

It appears these commands use an iterative algorithm

to find the date, since future dates take more processing time. This routine is looping over and over, figuring the day of the week by saying 'the day after Tuesday is Wednesday, the day after Wednesday is Thursday...' for several years. For absurdly futuristic dates, such as the year 65,000, this could take a long time. It also means your Amiga will run a tiny, tiny bit slower in the future.

Several Amiga owners report some commercial software disks have futuristic dates. It is easy to create a 'date virus'. In the CLI, if you type '-76' as the year part of the date for the DATE command, you have just set the date ahead to the year 2076.

If you thought it might set the date to 1976, remember that the earliest date possible is January 1, 1978. If you create a file on a disk now - just type 'DATE > now' - this disk has a 'date virus'.

To check a disk for files with futuristic date stamps, first reset the date to the present with 'DATE 13-Jan-86', for example. Then type 'LIST SINCE TODAY' at a CLI prompt. This will list every file in this directory that has a future date. In fact, the word 'Future' will appear in the date column of the LIST output.

To correct the date on a file, it must be copied or changed. Without removing or inserting any disks, you could COPY each and every file to another name, delete the old one, and rename it. This process is time consuming, to say the least. Also, COPY does not copy the comment made with the FILENOTE command, a serious flaw in itself.

Just as files have dates, directories have dates. A directory's date is reset when a file is copied into the

directory. To clean a directory node, you have to copy any file into that directory, and then delete the file.

If you don't remove EVERY file on a disk that has a bad date stamp, it doesn't remove the virus.

The key concept is **without inserting any other disks**. If you remove one disk to insert another, that new disk is validated. If the new disk has a futuristic date, the date is advanced. Is this a Catch-22? Yes, but you can always use the **RAM: disk**.

Alternatively, you can copy the files to the RAM: disk, and then back to the floppy. This might not be possible if the file is large, or if you have only 256 K of RAM. This method is a lot faster than disk-to-disk copies.

If you feel comfortable using the CLI, and have two drives, you can COPY the 'DATE' and 'COPY' commands to the RAM: disk, and 'CD' to the RAM: disk. Insert the offending disk in DFO:, and a new, formatted disk in the other drive. Change the date to the present day, and then 'COPY DFO: DF1: ALL'.

A much nicer method of changing a file's date is a program called 'touch'. The name is borrowed from a Unix utility that changes a file's date to the present. The source code for 'touch' is given in ABasic™ in Figure 1 and the Lattice C™ listing in Figure 3. See Figure 2 for an example session using the ABasic™ 'touch' program on a single drive Amiga, and Figure 4 to use the Lattice™ program.

These programs reset the date on a file by reading the first byte of the file, and then writing the same byte back. This doesn't damage the file in any way. Although the file is unchanged, it fools the operating system into changing the date stamp on the file.

This misfeature has serious implications for battery-backed clock users. Does this mean the system clock could still be wrong, even with a battery-backed clock? Yes, according to Harry Evangelou at Akron Systems Development, the makers of A-Time™, a battery-backed clock for the Amiga. Battery-backed clocks only reset the time on start-up, and the 'date virus' can reset the date any time after starting your Amiga.

Evangelou said he is considering methods to insure the system date stays correct, but at the expense of slowing down the Amiga. "Our clock chips are interrupt-driven, so we could generate interrupts every few seconds," and correct the system time when AmigaDOS set the clock too far into the future, he said.

Ultimately, the only sure cure for the 'date virus' is from Commodore-Amiga. According to a Usenet message from Commodore-Amiga representative Neil Katin, "the [AmigaDOS™ version] 1.1 release does not have the problem of a single file bumping the system date ahead... we fixed what, in retrospect, was obviously a bad effect... Sorry for the pain in [version] 1.0."

Meanwhile, with these 'touch' programs, you can remove the 'date virus' from your version 1.0 disks.

I will soon have a program that will remove the 'date virus' from every file on a disk, automatically. It will be placed in the public domain library at **Amazing Computing**, see our Public Domain Software article for details on obtaining **Amazing Computing** and **AMICUS** Public Domain disks.

Figure 1:

Simple 'touch' program in ABasic:

```
10 input "File to touch: ", f$
20 open "R", #1, f$, 1
30 field #1, 1 as b$
40 rget #1, 1
50 rput #1, 1
60 close #1
```

Figure 2:

An example session using ABasic™ 'touch', with one disk drive:

Start ABasic, and type or load the 'touch' program. After the ABasic starts, insert a disk that has the 'date virus'. To show that the date is corrupt, ask AmigaDOS™ for the date.

```
Ok shell "date"
Thursday 24-Oct-75 13:45
Ok
```

Reset the internal date to the present day.

```
Ok shell "date 12-Jan-86"
```

Run the 'touch' program for each file that has a bad date.

```
Ok run
File to touch: ? sample.text
Ok
```

Figure 3:

Simple 'touch' program in Lattice C:

```
#include "stdio.h"
#include "fcntl.h"

main(argc,argv)
int argc;
char **argv;
{
    int tfile; /* file handle for 'touch'ed file */
    char buf[4]; /* buffer to hold first character */

    /* If the user only typed 'touch' in the CLI, */
    /* explain how to use the program, and then exit. */

    if (argc==0) {
        puts("Usage: touch <filename>");
        puts("This program resets a file's date to the current date.");
        exit(1);
    }

    /* Otherwise, they supplied a filename */

    tfile = open( argv[1], O_RDWR );

    /* if filename doesn't exist, warn the user */
    if (tfile==EOF) {
        puts("That file does not exist.");
        exit(1);
    }

    /* Read the first character, then write it back, and
       close the file. */

    read( tfile, &buf, 1 );
    lseek( tfile, 0L, 0 );
    write( tfile, &buf, 1 );

    close( tfile );
}
```

Figure 4:

An example session using Lattice C 'touch' program:

After you have loaded the CLI, insert a disk that has the 'date virus'. Copy the CD, DATE and 'touch' programs to the RAM: disk, and CD to the RAM: disk.

```
1> copy sys:c/copy to ram:
1> copy sys:c/date to ram:
1> copy sys:c/cd to ram:
1> copy touch to ram:
1> cd ram:
```

To show that the date is corrupt, ask AmigaDOS™ for the date.

```
1> date
Thursday 24-Oct-75 13:45
```

Reset the internal date to the present day.

```
1> date 12-Jan-86
```

Type 'LIST SINCE TODAY' to find the files that have corrupt dates.

```
1> LIST SINCE TODAY
```

Run the 'touch' program on every file named in the LIST output.

```
1> touch sample.text
```

CD to every directory on this disk, and repeat the LIST SINCE TODAY and 'touch' sequence.

•AC•

EZ-TERM: Terminal Emulation Program

by Kelly Kauffman

Welcome to the wonderful world of ABasiC™! As you may, or may not know, ABasiC™ has a slight bug concerning the use of the Serial Port through ABasiC---they forgot to support it. Oh well, this isn't to point fingers or to assign blame, we will just work around it.

This is performed basically through "Poke"ing any characters you want sent out into memory, and "Peek"ing any incoming characters being sent to you. The backbone of the program lies in lines 1250 to 1470. This was taken from a programmer who obviously had a great deal of experience with the Amiga. However, I am sorry to say, I cannot find his name in my mess of papers scattered throughout the room. However, when I do find the name, I will let all know. (Sorry!!!)

The program itself is pretty much self-explanatory. It utilizes the "HELP" and the function keys (F1-F10). The help gives you a description of what each function key does. But you know what they say, experimentation is the best education.

The Xmodem send/receive part of the program was originally hacked by myself. However, I had many complaints that it would not work 100% (I had no problem with it). So, when I saw a program on a BBS, named "SuperXmodem," I decided it must be the best. So I downloaded it, (using the Xmodem!), and worked it into my program. The credit for the Xmodem send and receive part of my program must now go to K. L. Colclasure.

When you first type "RUN" and press your return key, a small window will open up on the bottom of the screen with a quick-reference to the function keys in it. A large window will open, filling the rest of the screen. This large window is where all of your incoming and outgoing data will be displayed. To get the program running completely though, you must move your mouse anywhere in the large window and click the left button. Otherwise, you will only be able to see what is coming in, and you will not be able to reply. Once you do this, you are running the program at 100% capacity.

Now, to review some of the features of EZ-TERM in a little more detail:

The CAPTURE feature: F1

Activated through the F1 key. When this key is pressed, it toggles the **capture** between **on** and **off**. If the capture is on, a small message will appear in the lower-right hand corner of the screen telling you so.

Use capture to save incoming text into a buffer for later reviewing and/or saving for later use.

Use F3 to **Save** the buffer off to a filename of your choice and/or use F4 to **Review** the buffer on the screen.

The LINEFEED TOGGLE feature: F2

Depending on what type of system you are calling, some BBS' do not send the LineFeed character which is needed to scroll the text up the screen. If you are not receiving the LineFeed (LF) character, you will know it right away...everything will just type over itself. Turn the Linefeed Toggle ON.

When turned on, the LF Toggle feature will automatically scroll the screen up, thus eliminating the need for the LF character.

If everything you are receiving is double-spaced, turn the LF Toggle feature OFF. This should return your incoming data to single-spacing. If it doesn't, then you are being sent data double spaced.

The SAVE feature: F3

The **SAVE** feature saves the contents of the Capture buffer to a file of your choice.

You can specify any filename, including **PRT:** to send all data to the printer. For more information on filenames, etc., consult your owners manual.

When you save your buffer, it **DOES NOT** clear out and erase your capture buffer, to do this, use **F8**.

The REVIEW feature: F4

The **REVIEW** feature lets you see what is contained in your capture buffer.

Simply press the **F4** key. If you wish to stop the review, hit any key.

The LOAD BUFFER feature: F5

The **LOAD BUFFER** feature lets you load data from a file, into your capture buffer. NOTE: This feature does not merge the data (i.e. retain what is in the capture buffer already and tack the new information onto the end). It instead, clears out all of the old information and loads in the new.

Abort the Load feature and retain your current buffer by just hitting return at the request for the filename to load.

The UPLOAD feature: F6

Use the **UPLOAD** feature to send the contents of your capture buffer to a computer with which you are connected.

The business about "Use Prompts ?" pertains primarily to Compuserve, although I have seen other private bulletin boards use this feature. What it means is, on BASIC FILES ONLY it will send one line of a basic program and wait for your connection to send you a "prompt" character back to let the Amiga know it is ready to receive the next line.

The prompt character used in this program is the ">" character. To change this, change the > in line **2520** to any character you wish.

The part about sending a return after each line is for use especially on Compuserve. If you reply to this "No," it will just send the LineFeed character to the connection. This is not good enough for Compuserve, it needs the RETURN character.

The DUPLEX TOGGLE feature: F7

This feature toggles the Duplex on and off each time you press **F7**. The Duplex should be ON if you cannot see what you are typing. This will "echo" all of the characters you type onto the screen immediately. If you are getting double vision on what you are typing, make sure the Duplex is off, this should correct your problem.

The CLEAR BUFFER feature: F8

Use this option to wipe-out all information contained in the capture buffer to make way for any new data.

The XMODEM RECEIVE feature: F9

Use this feature to receive a file using the Xmodem Protocol. The Xmodem protocol will provide for error-free transmission of files between your Amiga and your connection. Reply to the filename question to have the file saved under a file of your choice.

The XMODEM SEND feature: F10

Use this feature to send a file using the Xmodem Protocol. This will allow you to send a file to your connection, again, error-free. Reply to the filename question to specify which file you wish to send.

Additional Notes:

To exit EZ-TERM, hold down the ALT key and press the "Q" key.

To send control characters, hold down control and press whatever key you want sent. NOTE: If you want to send a CTRL-C, you MUST hold down the ALT key as well as the CTRL key, then press "C." Otherwise, you will "Break" out of the program!

It is recommended to Xmodem RECEIVE to the RAMdisk. It is faster and cuts down on connect time.

This program was not intended for the first-day computer owner. You should have a good working knowledge of the Amiga file structure system to use the send and receive aspects of the program to their full potential.

Please feel free to modify and/or distribute this program to friends, neighbors, stores etc., but PLEASE leave my name on it as I have put in some time to get this thing working!!!

Thanks....and Enjoy!!!

EZ-TERM PROGRAM

by
Kelly Kauffman

(Note EZ-TERM is available on Amicus PDS#2 disk)

```
1000 'EZ-TERM V.2.19 - - - - - 12/08/85
1010 '
1020 ' by Kelly Kauffman
1030 '
1040 ' Xmodem Receive and Send were taken from a
      program written by
1050 ' K. L. Colclasure ----->>>>>>Thanx
1060 ' Program only supports 300 baud due to how
      sslloowwww ABASIC is.
1070 ' Have fun!
1080 '
1090 graphic(0):screen 1,1,0:gosub 2510
1100 print "ALT+Q - Quit Program"
1110 dim a$(10000)
1120 rem Kelly Kauffman,3170 Sprout
      Way,Sparks,NV, 89431,CIS[70206,640]
1130 rgb 0,0,0,10:rgb 1,12,12,12:rgb
      3,0,0,15:rgb 2,12,0,4
1140 "???"
1150 print spc(1);inverse(1); "To send Control
      Characters, hold down ALT & CTRL, and press
      the ctrl key." ;inverse(0): gosub 1250
1160 print:print
1170 print "EZ-TERMINAL VER2.21"
1175 print "by Kelly Kauffman"
1180 on error goto 3790
1190 "???" FOR PUBLIC USE ONLY"
1200 "???"spc(7);"Put mouse in this window, and
      click LEFT button to get rollin"
1210 "???" [HELP] is available."
1220 get char$
1230 if char$<>"" then if asc(char$)=155 then
      gosub 1470
1240 gosub 1340:gosub 1400:print char$;
      goto 1220
1250 baud%=300
1260 iobase%=&hfff000
1270 serdatr%=&h18+iobase%
1280 serdat%=&h30+iobase%
1290 serper%=&h32+iobase%
1300 intreq%=&h9c+iobase%
1310 poke_w serper%,(1/baud%)/(.2794*1e-06)
1320 return
1330 'write
1340 if char$="" then return
```

```

1350 if asc(char$)=241 or asc(char$)=209 then
2640
1360 poke_w serdat$,asc(char$)+256
1370 if plex=1 then gosub 3830
1380 return
1390 'read
1400 char%=peek_w(serdatr%)
1410 if (char% and 16384) = 0 then char$="":
return
1420 if len(buff$)>253 then a$(num)=buff$:
num=num+1:buff$=""
1430 char$=chr$(char% and 127):poke intreq$,8:
if cap=1 then buff$=buff$+char$
1440 if asc(char$)=13 and lf=1 then ?:if cap=1 then
buff$=buff$+chr$(10)
1450 return
1460 gosub 1390: print char$,:goto 1460
1470 get char$
1480 if char$="" then return
1490 if asc(char$)<48 or asc(char$)>63 then
return
1500 if asc(char$)=48 and cap=1 then cap=0:
num=num+1: a$(num)=buff$:buff$="":
get char$: get char$:gosub 2720:return
1510 if asc(char$)=48 and cap=0 then cap=1:
gosub 2660
1520 if asc(char$)=48 then get char$:get
char$:a$(num)=buff$:buff$="": return
1530 if asc(char$)=49 and lf=0 then lf=1:?:?"
Simulated Linefeeds ON": getchar$:
get char$: return
1540 if asc(char$)=49 and lf=1 then lf=0:?:
?"Simulated Linefeeds OFF":get char$:
get char$:return
1550 if asc(char$)=50 and buff$="" and num=0 then
?:print " BUT THERE'S NOTHING TO
SAVE!!!"
1560 if asc(char$)=50 and len(buff$)=0 and num=0
then get char$: get char$:return
1570 if asc(char$)=50 then cap=0:num=num+1:
a$(num)=buff$:buff$=""
1580 if asc(char$)=50 then get car$: get car$:
input "Save as ->": file$
1590 if asc(char$)=50 then if file$="" then
?"Aborted.": get a$: get a$: return
else open "o",#4,file$
1600 if asc(char$)=50 then for q=0 to num:
? #4,a$(q),: next q:close #4:
?" CAPTURE BUFFER SAVED"
1610 if asc(char$)=51 then num=num+1:
a$(num)=buff$:buff$=""
1620 if asc(char$)=51 then gosub 3690:return

```

```

1630 gosub 1660
1640 get char$:get char$
1650 return
1660 rem
1670 if asc(char$)=53 then gosub 1960:return
1680 if asc(char$)=52 then gosub 1750:return
1690 if asc(char$)=54 then gosub 2190:return
1700 if asc(char$)=63 then gosub 2220:return
1710 if asc(char$)=55 then gosub 2400:return
1720 if asc(char$)=56 then gosub 3660:
gosub 2860: get char$: get char$: return
1730 if asc(char$)=57 then gosub 3660:
gosub 3350: get char$: get char$: return
1740 return
1750 scncrlr
1760 get char$:get char$
1770 rem
1780 input "Load what Filename ----->":file$
1790 if file$="" then print "Aborted.":goto 1930
1800 erase a$:dim a$(6000)
1810 x=instr(1,file$,".bas"):
y=instr(1,file$,".BAS"):
if x=0 and y=0 then bas=0 else bas=1
1820 close #4
1830 num=0
1840 open "i",#4,file$
1850 on error goto 3790
1860 if bas=1 then line input #4,buff$
else get #4,w$
1870 if bas<>1 and w$="" then 1920
1880 if bas=1 then a$(num)=buff$
else b=len(a$(num)): if b>253 then
num=num+1: a$(num)=a$(num)+w$
1890 if bas<>1 and b<=253 then
a$(num)=a$(num)+w$
1900 if bas=1 then num=num+1:buff$=""
1910 if not eof(4) then 1860
1920 print "Complete File Loaded."
1930 on error goto 3790
1940 close #4
1950 return
1960 scncrlr
1970 print
1980 get char$:get char$
1990 if bas=1 then input "Do you want to use
prompts";yn$: if yn$="n" or yn$="N" then
prompt=0 else prompt=1
2000 print :print "Do you want a return sent after
each line": input yn$: ifyn$="Y" or yn$="y"
then retn=1 else retn=0
2010 scncrlr
2020 print " Beginning Upload": gosub 2790

```

```

2030 get char$: get char$
2040 for i=0 to num
2050 for q=1 to len(a$(i))
2060 qwer=asc(mid$(a$(i),q,1)): if bas=1 and
qwer=10 then qwer=13
2070 get char$:if char$<>"" then print:?"UPLOAD
ABORTED BYUSER": print: print: gosub 2830:
goto 2180
2080 poke_w serdat%,qwer+256
2090 print mid$(a$(i),q,1);
2100 if baud%=300 then sleep 13000
2110 next q
2120 if retn=1 then poke_w serdat%,269:print
2130 if prompt=1 then gosub 2480
2140 next i
2150 print: print
2160 print "Buffer Upload Complete."
2170 gosub 2830
2180 return
2190 if plex=1 then plex=0:?:
print "    FULL DUPLEX":
goto 2210
2200 if plex=0 then plex=1:?:
print "    HALF DUPLEX"
2210 get char$: get char$: return
2220 scnc!r
2230 print "F1 - Toggles Capture on/off to capture
incoming data and save in the buffer."
2240 print "F2 - Toggles Simulated LineFeeds On/Off.
Turn on if you get type-overs."
2250 print "F3 - Save Capture Buffer. Saves the
contents of the buffer to a file of your choice."
2260 print "F4 - Review Buffer. Lets you see the
contents of the buffer. Press any key during the
review to abort."
2270 ?"F5 - Load Buffer. This will load a file of your
choice into the Buffer."
2280 ?" NOTE: It does NOT merge the data, it clears
out the old info then loads the new."
2290 print "F6 - Upload Buffer. The question about
prompts means that the Amiga will"
2300 print " wait for the computer you are hooked
up with sends a prompt character."
2310 print " This character is the "">"" character.
If you are sending other than a"
2320 print " source code, reply ""N""o to this
question, then the Amiga will just"
2330 print " upload the entire buffer without
waiting for prompts."
2340 ? "F7 - Duplex Toggle. Toggles between Full
and Half duplex for seeing what you are typing."

```

```

2350 print "F8 - Clear buffer. Clears out the
capture/upload/review buffer."
2360 print "F9 - Xmodem Receive."
2370 print "F10- Xmodem Send."
2380 print "ALT + Q - Quit Program."
2390 return
2400 print
2410 print
2420 print "          BUFFER CLEARED"
2430 erase a$
2440 dim a$(6000)
2450 buff$=""
2460 num=0
2470 return
2480 gosub 1390
2490 if char$<>">" then 2480
2500 return
2510 rem draw windows for Function definitions
2520 window #1,0,200,640,100,"Function Key
Definitions"
2530 cmd 1
2540 ? inverse(0);"Cap Off";
2550 print inverse(0);" ";inverse(1);"F3";
inverse(0);
2560 print "-Save "; inverse(1); "F4"; inverse(0);
"-Rvw ";inverse(1);"F5";inverse(0);
2570 print "-Load "; inverse(1); "F6"; inverse(0);
"-Upload ";inverse(1);"F7";
2580 print inverse(0);"-Duplex "; inverse(1);
"F8"; inverse(0);"-Clear ";
2590 print inverse(1);"F9";inverse(0);"-XRec
inverse(1); "F1"; inverse(0); "0-XSnd";
2600 print at(71,0);
2610 window #2,0,0,640,186," EZ-TERM V2.19
----->by Kelly
Kauffman<===== "
2620 cmd 2
2630 return
2640 close #1,2
2650 end
2660 cmd #1
2670 rem capture On
2680 print at(0,0);inverse(1);"Cap On";
inverse(0);" ";
2690 print at(71,0);
2700 rgb 0,10,0,0
2710 cmd #2:return
2720 rem
2730 cmd #1
2740 rem capture off
2750 print at(0,0);inverse(0);"Cap Off";
2760 print at(71,0);

```

A GREAT COMPUTER... OUTSTANDING SOFTWARE... ...AND INCREDIBLE PRICES!

There may not be a better personal computer than the Commodore Amiga. But no computer can be better than the software that runs on it. Micro-Systems Software, makers of OnLine! and BBS-PC for the Amiga, proudly announce another link in their chain of value-packed software.

Analyze! is a powerful electronic spreadsheet program. Essentially, this program is a full-screen calculator where you can organize your data into rows and columns. These rows and columns can be analyzed with simple mathematics or complicated formulas. Rows and columns can be duplicated to avoid re-typing. Both data and formulas can be edited with only a few keystrokes.

From home budgets and check registers to financial modeling and your company's general ledger, all manner of bookkeeping tasks become faster and easier with Analyze! An outstanding value at only \$99!

OnLine! combines features and convenience in a high quality package that will meet all your telecommunications needs. With OnLine!, you can use your Amiga as a window to the world of information that is just on the other side of your telephone. You can link up with commercial information services for stock quotes, airline information and reservations, technical databases, and thousands of other business and entertainment tasks. You can also plug into local bulletin board systems (BBS) and discover a new world of information and software for your computer. Corporate users can use OnLine! to let their Amiga access data stored on the company's mainframe computer.

OnLine! is the finest program of its type available for the Commodore Amiga. You can't lose when you get "online" with OnLine!. All for a down to earth price of only \$69!

2400 bps modems! 2400 bps modems are breaking the speed barrier in telecommunications, and Micro-Systems is breaking the price barrier in 2400 bps modems. Transfer files 2 times faster than a 1200 bps modem and 8 times faster than a 300 bps modem. Micro-Systems will sell you a Hayes Smartmodem compatible 2400 bps modem, a special Amiga serial cable, and a copy of OnLine!, all for \$429.

That's right, the modem, the cable, and the software, all you need to begin using your Amiga as a terminal to the world, priced at \$429! Hundreds less than our competition!

Micro-Systems Software, Inc.

4301-18 Oak Circle
Boca Raton, FL 33431

(800) 327-8724/National, (305) 391-5077/Florida

Ask us about our Amiga bulletin board program, BBS-PC.
The first BBS for Amiga!

AMIGA, OnLine!, Analyze!, BBS-PC, and Smartmodem are trademarks of Commodore-Amiga, Inc., Micro-Systems Software, Inc., and Hayes Microcomputer Products, Inc., respectively.

```
2770 rgb 0,0,0,10
2780 cmd #2:return
2790 cmd #1
2800 print at(35,0);inverse(1);"Upload";
2810 print at(71,0);
2820 cmd 2:return
2830 cmd #1
2840 print at(35,0);inverse(0);"Upload";
      at (78,0);
2850 cmd 2:return
2860 goto 3170:' XMODEM RECEIVE & SEND
```

```
-----
2870 size% = 5: timeout% = 500: baud% = 300
2880 gosub 3660: goto 3100
2890 key% = asc(key$): poke_w out%, key% + 256
2900 if key% = nak$ then sleep 35000
2910 return
2920 gflag% = 0: char% = peek_w(in%)
2930 if (char% and 16384) = 0 then return
      else gflag% = -1
2940 char% = char% and 127: poke intrq%, 8:
      return
2950 t = 0: toflag% = 0
2960 char% = peek_w(in%)
2970 if (char% and 16384) = 0 then t = t + 1
      else 3000
2980 if t > timeout% then toflag% = -1: return
2990 goto 2960
3000 char% = char% and 255: poke intrq%, 8:
      return
3010 cksum% = 0: for i = 1 to 131
3020 cksum% = (cksum% + buf%(n,i)) and 255:
      next i
3030 if cksum% = buf%(n,132) then 3050
3040 sleep 10000:print spc(20);"Cksum error in";
      blk%: key$ = nak$: return
3050 mblk% = blk% and 255
3060 if mblk% = buf%(n,2) then 3080
3070 print spc(20);"Sync error in"; blk%: key$ =
      nak$: return
3080 blk% = blk% + 1: n = n + 1: key$ = ack$
3090 print spc(20);" Received"; blk% - 1;
      chr$(13);: return
3100 goto 1250
3110 print: on error goto 0
3120 get key$: if key$ = "" then 3150
3130 if key$ = chr$(155) then 3650
3140 gosub 2890
3150 gosub 2920: if (gflag%) then print
      chr$(char%);
3160 goto 3120
```

```

3170 get char$:get char$:print: print "Xmodem
    Receive, enter filename: ";
3180 line input file$: if file$ = "" then
    ?" Aborted.": ? : ? : goto 3100
3190 open "o",#5,file$:rem close #5
3200 blk% = 1: n = 1: eotflag% = 0: key$ = nak$
3210 gosub 2890: for i = 1 to 132: gosub 2950
3220 if (toflag%) then key$ = nak$: goto 3210
3230 if (i = 1) and (char% = eot) then 3260
3240 buf%(n,i) = char%: next i: gosub 3010
3250 if n > top% then 3270 else 3210
3260 eotflag% = -1
3270 rem open "a",#5,file$
3280 for x = 1 to (n - 1): for y = 4 to 131
3290 print #5, chr$(buf%(x,y));
3300 next y,x
3310 Rem close #5
3320 if not (eotflag%) then n = 1: goto 3210
3330 gosub 2890
3340 close #5:print: print "Transfer complete ...":
    goto 3100
3350 get char$:get char$:print: print "Xmodem Send,
    enter filename: ";
3360 line input file$: if file$ = "" then ?" Aborted.":
    ? : ? : ? : goto 3100
3370 on error goto 0
3380 open "i",#5,file$
3390 on error goto 0
3400 n = lof(5): n = n / 128
3410 if int(n) < n then n = int(n) + 1 else n = int(n)
3420 print "File open, ";n;"records."
3430 n = 1: blk% = 1
3440 buf%(n,1) = soh: buf%(n,2) = blk% and 255
3450 buf%(n,3) = buf%(n,2) xor 255
3460 for i = 4 to 131
3470 if not eof(5) then get #5, char$ else
    char$ = eof$
3480 buf%(n,i) = asc(char$): next i
3490 gosub 3560: gosub 3590: print spc(20);
    " Sent block"; blk%: chr$(13);
3500 if not eof(5) then blk% = blk% + 1: goto 3440
3510 close #5
3520 key$ = chr$(eot): gosub 2890
3530 gosub 2920: if not gflag% then 3530
3540 if char% = nak then 3520
3550 if char% = ack then 3340 else 3530
3560 cksum% = 0: for i = 1 to 131
3570 cksum% = (cksum% + buf%(n,i)) and 255:
    next i
3580 buf%(n,132) = cksum%: return
3590 if blk% = 1 then 3620
3600 for i = 1 to 132: key$ = chr$(buf%(n,i))

```

```

3610 gosub 2890: sleep 30000: next i
3620 gosub 2920: if not gflag% then 3620
3630 if char% = nak then 3600
3640 if char% = ack then return else 3620
3650 get key$: if key$ = "" then 3120 else
    fkey% = asc(key$)
3660 ack$=chr$(6): nak$=chr$(21):eot$=chr$(4):
    ack=6:nak=21: eot=4:soh=1: eof$=chr$(26):
    timeout%=500:size%=5
3670 erase buf%
3680 option base 1:dim buf%(size%*8,132):
    top%=size%*8: baudr%=&hdf032:
    out%=&hdf030: in%=&hd f018:
    intrq%=&hdf09c: return
3690 scncir:print:print
3700 get char$:get char$
3710 for q=0 to num
3720 if bas=1 then print a$(q)
3730 if bas=0 then print a$(q);
3740 get char$
3750 if char$ <> "" then ? : ? : ? : ? "Buffer Review
    Aborted.": get char$:return
3760 next q
3770 ? : ? : ? : ? "Buffer Review Completed."
3780 return
3790 print "----->>>";err$(err);"<<<==== at
    line "; erl
3800 print:print
3810 on error goto 3790
3820 resume 1330
3830 print char$;
3840 if cap=1 then buff$=buff$+char$
3850 if asc(char$)=13 then ?
3860 return

```

•AC•

Live to Compute!
Please
Don't Drink
&
Commute!

ROOMERS

by
The Amigo

Welcome to "Roomers". Ok, so it's a dumb name. However, in this column we will be giving you the latest low-down on what is happening on the Amiga scene, usually before it happens. We may be off base in a few cases, but we will try to be on as much as we can. So let's start.

We got books.

Really we do not have them yet, but we should soon.

Bantam Books is publishing the three **Amigados manuals**, available now at \$50 for the set. Bantam's price will be \$24.95 and will be available by March.

Sybex has two books in the works: **The Official Amiga Manual** and the **Amiga Developer's Manual**. Sources say that the Developer's manual will be about 400 pages and be out around February.

Meanwhile, the new **Amiga ROM Kernel Manual** is days away from being shipped. Its more than 1600 pages cover almost everything about the insides of the Amiga, plus sections on the IFF "picture standard". IFF was developed by Electronic Arts with the Amiga stamp of approval, and is used by the utilities such as **Deluxe Paint™**, **Graphicraft™**, etc.

Software:

Software is the big news of the month (and probably will be for some time). It looks like that "avalanche" of Amiga software is about to hit the streets.

Electronic Arts is shipping **Deluxe Paint™**, **Dr. J. and Larry Bird One on One™**, **Archon™**,

Financial Cookbook™, and **The Seven Cities of Gold™**.

All the Electronic Arts software is copyprotected. EA explained that the software is late due to some last-minute changes they had to make when version 1.1 of the Amiga Operating System was introduced.

OS9™, a Unix (TM Bell Labs) lookalike should be ready sometime this spring. OS9 was originally written for the 6809 and boasts a large user base.

Editors:

In editors, **MicroEmacs** has been ported to the Amiga and is available now as public-domain software.

LSE™, the **Lattice Screen Editor™**, is now shipping from **Lattice** (who else). Reports are that the first version is a little buggy, and a new version will be out by the time you read this. Check the program before you buy. If you can type upper case F, you have a good version. The early version would not shift the f key.

Languages:

Lattice is now shipping V3.03 of their compiler. Version 3.04 should have both **IEEE** and **Motorola Fast Floating Point** support.

Manx should have **Aztec C™** out this month.

Borland's TurboPascal™ for the Amiga is not at beta test yet.

A Washington-based company is about to unleash its version of a **Modular-2** compiler any day now.

Data Communication Packages:

Everyone and his sister is working on Data Communications packages. You can go to any dealer and pick up **Maxicom**™, **Elterm**™, and **Online**™ all vt100 compatible with upload and download capabilities. Many more are on the way.

What we need is a local area network that talks through the Amiga's serial port, at **MIDI** (31.5kbaud) speeds! Atari is hard at work on using the MIDI port for the ST already. But please, no more modem programs.

Bug Report:

There is a bug in the Amiga software, both in version 1.0 and 1.1. Seems that when you close a font, the font really stays around (if you need to open it again, the system will not read from the disk). Well, when the memory gets low and an application requires more room, it grabs the space taken by the font but does not erase the pointer to the font. Later, when the application goes to use the font again, the pointer is still there, and an index into the (not-available) font crashes the machine. This will be fixed in V1.2 (no word yet on its release date ...stay tuned!)

Electronic Arts has supposedly reverse-engineered **AmigaDOS**™ and rewritten it to couple it closer to the hardware capabilities. I do not know if it will be made available, or when, but I would like to see it, as **AmigaDOS**™ is pretty slow.

Hardware:

In the glorious world of hardware, my crystal ball tells me those busy folks at **Amiga** are working on a

68020 version. Code named "**Ranger**", details are a little too sketchy to go into, except that it will be 100% compatible with the current Amiga (model number A1000, you know), better monitor support (400 lines non-interlaced) and will have a built-in hard disk. Amiga is badgering their developers to stay away from 68000-specific coding practices to be compatible with the new generation of machines, and rewrote parts of the Kernel to support 68010 and 68020 interrupt handling.

As for current Amigas, they are shipping with a new rev of the display chip that allows for half-intensity mode. If IBM can claim their graphics adapter allows for 16 colors (really 8 with a half-intensity option), the Amiga can claim 64 colors at once, rather than 32.

CSA showed a **68020 board** at **Comdex** that you can add on to your Amiga; they are already working on a new model that runs at more than 14MHz...

Okidata is offering its **Okimate-20**™ owners a new chip that eliminates the streaking caused when used with the Amiga. Call 1-800-OKI-DATA for details.

Tecmar is shipping its **T-card**™ and **T-disk**™, with the **T-modem**™ to follow in February. Tecmar raised its prices at Comdex. It appears they believe people are starved for add-ons and will pay any price for peripherals. We'll see. Beta-testers have told me the **T-disk**™ is only twice as fast as the 880K "stiffies" that comes with the Amiga, the fan sounds like a vacuum cleaner, and the hardware does not configure on startup.

(Ed-note: all comments made on Beta-test equipment or software must be labeled carefully, the items are

✓AMIGA \$9.95 ea.

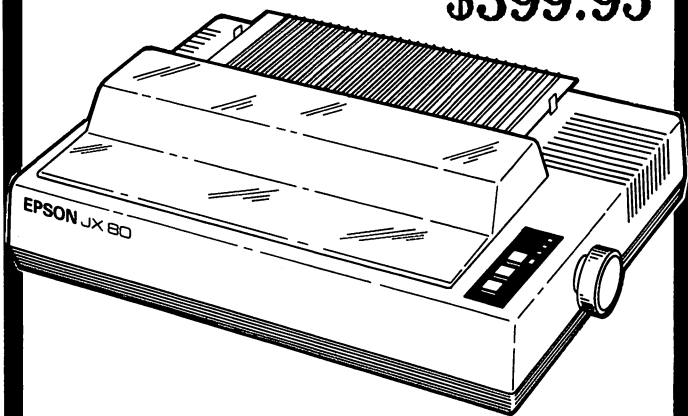
Public Domain Disks

Disk 1 Star Trek (Text), EZ-Terminal,
Amiga Doodle

Disk 2 Copperstate Copier for 2 drives

Disk 3 Amiga Info

EPSON® JX-80 Color Printer **\$399.95**

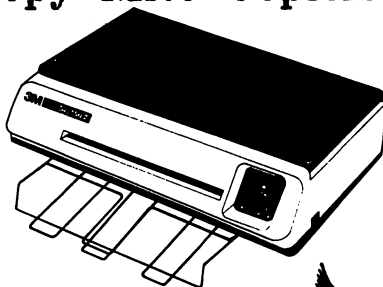


Amiga Printer Cables \$29.95

3M Copy Mite Copiers

Copy Mite 1
\$169.95

Copy Mite 2
\$269.95



Cardinal Software
13646 Jeff Davis Hwy.
Woodbridge, VA 22191



Order NOW! 800 762-5645

Info (703) 491-6502 SHIPPING EXTRA

being tested for the first time by third party individuals and should not fall under the same scrutiny as a finished product. The developers are attempting to remove all complaints at Beta-test.)

CardCo, a name C64 owners are sure to remember, is about to unveil their **1MB** add-on board for the Amiga. It will be priced below \$300, and is currently pending FCC approval.

Akron Systems Development is shipping **A-time™**, a battery-backed clock that sits on your printer port. The printer plugs into A-time™ and the clock is transparent to the printer. Akron is rumored to be ready to announce a **2MB** board any day now for under \$1000, upgradable to **8MB** in the same box when **1MB** chips become available.

Those marketing folks at **Commodore** have decided to change the name of the frame grabber to **Amiga LIVE!™**, and have it available before the end of February. **A-squared**, the manufacturer, has units in beta-test now.

User Groups are popping up everywhere. Look around Rutgers (NJ), Boston (Boston Computer Society), Florida and North Carolina, with more added all the time.

Hey, That's all for now. I am following up on more leads for next month. If I forgot you, it is because you have not told me yet!

•AC•

Miga-Mania

by
Perry Kivolowitz

Welcome to "Miga-Mania," a monthly column to serve as a grab bag for useful information for using, programming, and enjoying the AMIGA™ personal computer. Please use this column as a resource for information which, otherwise, will not fit into separate neat categories.

For the first few issues, I will populate the column with quickies and experiences which I have had and might help you. But, be warned! My bent is hardcore internals, so much of what I will have to offer will be technical in nature and oriented toward advanced users and or developers.

For instance, I can easily tell you how to dynamically generate pretty looking menus and such, but I still do not know what half the AmigaDOS™ commands do! One of the truly beautiful things about starting fresh (as Amazing Computing is) is we have no precedents or guidelines to follow.

What would YOU like in this column?

I would like to cover such topics as:

- * General Programming Tips - not particular to the AMIGA or even to a specific language, just general techniques to assist persons in developing quality code.
- * Programming Quickies and Suggestions - Specifically for the AMIGA. Here, I'd like to split time evenly between C and Basic but, of course, any language you use is fair game here.

- * Developing Neat or Effective User Interfaces utilizing Intuition, another detailed programming topic which deserves a forum for discussion.
- * CLI - how to effectively navigate the "icky" CLI interface. (I include the word "icky" to demonstrate that you are allowed opinions here - Amazing Computing exists as a service to you, the user - we do not derive benefit from calling all that smells, a rose.)
- * Using the WorkBench - I had my machine three months before someone showed me how to keep a "cleaned-up" WorkBench window "cleaned-up." As you discover any short-cuts to agony or ecstasy, send'em here!
- * Using any of various packages or commercially available applications. Same as with the WorkBench, anything which will save you time, help you work more effectively and/or prevent a disaster, is fair game here.

I am looking to you readers and users to provide a steady stream of questions or contributions. The success of this column as a forum for the dissemination of useful AMIGA information depends upon **YOU!** Send your contributions to:

Usenet: ihnp4!ptsfa!well!perry

Compuserve: Don Hicks 76714,2404

USMail: Miga-Mania
c/o Amazing Computing
PiM PUBLICATIONS, Inc.
P.O. Box 869
Fall River, Ma. 02722

In the future, we may add further electronic means of submitting material besides Usenet (Unix to Unix network) and Compuserve.

General Helpful Advice #1

Second MicroFloppy Uses Memory!!!

Many people, who have not purchased the additional 256K ram expansion but have purchased the external 3.5" microfloppy drive, can not run many of the demonstration programs available from various sources. Try disconnecting the external floppy drive before booting your AMIGA. AMIGADos™ allocates additional disk buffers when it finds you have a second drive.

On a 256K machine the space used by the extra buffers may mean the difference between seeing the Mandrill or being treated to AMIGA_FIREWORKS_MODE. I have heard, Commodore (rightly) suggests the dealers push ram upgrades before they sell a user a second floppy. But...

General Helpful Advice #2

Mouse Traction

The mouse is a wonderful input device which is natural and easy to use for us humans. However, the poor critters are often asked to scurry over surfaces that are perhaps too smooth or too dirty.

If your work surface is too smooth (too glossy?), you will loose traction when not enough friction is developed between the work surface and the mouse's

roller.

If your work surface is dirty, gunk will eventually clog up the bearings inside the mouse which translate roller motion into x and y motion.

In either case, a mouse substrate (I can not say mouse-mat, it is a trademark of American Covers, Inc.) can really help by providing a clean, friction filled experience for your little friend.

Mouse pads are between six and twelve dollars in most computer stores and are well worth the money. However, you can go into most any home supply store and get a small square of 1/8 inch thick cork for less than a buck and it will do as good a job. (Or find a diving gear shop and ask if they sell one sided material for diving suits in small pieces. It is more expensive than the cork, but it has no cork crumbs or dust....Ed note.)

Intuition #1

Avoiding Falling Behind IntuiMessages

Intuition is the paradigm under which programs interface with the AMIGA's screen and windowing capabilities. When **Intui** wants your task's attention, it sends you an **IntuiMessage**, if you have enabled the IDCM (the "Intuition Direct Communications Message Ports"). Examples of IntuiMessages include those of class:

CLOSEWINDOW - Intuition's way of telling you that the user has clicked the window close gadget.

MENUPICK - The user has just selected an item from one of your menus.

NEWSIZE - Your window has just changed sizes due to the user dragging around the window sizing gadget (don't gadgets and windows get hurt by all that dragging?).

Every message your program receives (via GetMsg) **MUST** be replied to (via ReplyMsg). Bad things happen if you do not reply to a message sent to you. For one thing, messages occupy memory. If a message is left with out a reply, it is possible the memory it sits in will become lost to the system. More important, some IntuiMessages are the means of synchronization between your process and the others sharing the same screen. Delaying the reply of a synchronization message (the "verify" message classes) can cause other tasks to operate more slowly.

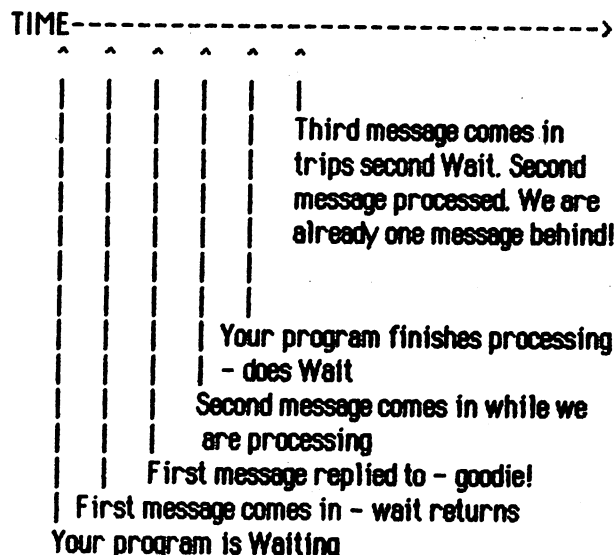
You might think a reasonable way to handle messages is the following:

```
while (there's more to do) {
    Wait (for a message);
    get the message;
    copy relevant information out to save area;
    reply to the message;
    process the message, maybe terminating this loop;
}
```

The good points of the above pseudo-code are that every message is promptly replied to and we reply to a message only when we are sure we have received one (the Wait does not return until you really do have a message waiting for you).

The bad point is that it would be very easy to fall behind in your message processing. The result of falling behind is things will stop happening when the user requests them and will **ONLY** happen if the user

punches a few more buttons, generating more messages and causing you to fall even further behind. For example:



Yes, the ROM Kernel Manual (RKM) says that Wait will return immediately if what you are waiting for has already happened, but my experience has shown the above can and does happen. (Use a loop patterned after the above pseudo-code to handle menu selections and then do a drum roll on the menu select mouse key, you'll see).

Instead of the above loop, do the following:

```
while (there is more to do)
{
    try to get a message;
    if (there wasn't a message)
    {
        Wait(for a message);
        go back to top of loop;
    }
    copy out relevant information to save area
    reply to the message
    process message;
}
```

This way, you are not forcing a wait prior to handling each message. As long as messages are queued for you, you will keep grabbing as soon as you can. An (untried) alternative is the **WaitPort** exec call. The WaitPort function documentation indicates it will perform the same checks and wait only if no message is

currently available.

Programming Quickly *1

Centralized Closing of Allocated Resources

Much of the code I have seen from various places, including Commodore itself, exhibits a common characteristic. As resources (such as device drivers, fonts, libraries, etc) are allocated the code to deallocate them, in the event the current allocation fails, is duplicated again and again. For example:

```
try to open resource 1
if failure then exit
try to open resource 2
if failure then
    close resource 1
    exit
endif
try to open resource 3
if failure then
    close resource 1
    close resource 2
    exit
endif
```

and so on...each time a resource is allocated the failure contingency code grows.

Instead of doing things this way, why not declare a flag word to keep track of which resources have been successfully allocated. In the event of a program error or even its normal termination, call a single routine which interrogates the flag word closing each resource in turn.

In C, this might be done as follows:

(text inside pairs of ``/*" and ``*/" are comments in C)

```
unsigned int resource_mask = 0;
```

```
/*define RESOURCE_11 /* if bit zero of
    resource_mask is on, then this resource has
    been allocated and must be deallocated
    sometime in the future.*/
```

```
/*define RESOURCE_2 2
    /* next bit for next resource */
```

```
/*define RESOURCE_3 4
    /* next bit for next resource */
```

```
/*define RESOURCE_4 8
    /* and so on */
```

The allocation code would look like:

```
try to allocate resource 1
if failure exit
resource_mask |= RESOURCE_1 /* this means
    ``or" in the value of RESOURCE_1 into
    the bits which comprise
    resource_mask.*/
```

```
try to allocate resource 2
if failure then call closer /* the closer will look at
    bits set in resource_mask and close or
    deallocate resources*/
```

```
resource_mask |= RESOURCE_2
try to allocate resource 3
if failure then call closer
resource_mask |= RESOURCE_3
```

and so on...

Lastly the closing routine might look something like...

```
closer()
{
    if RESOURCE_1 bit set in mask then deallocate
        resource 1
    if RESOURCE_2 bit set in mask then deallocate
        resource 2
    if RESOURCE_3 bit set in mask then deallocate
        resource 3
    and so on...
}
```

What do you think?

Well, that is all for this month. Please, feel free to comment on any current or future item from this column by referencing its topic heading as well as its number. You people have a wonderful opportunity to become immortal by being among the first to contribute to this column. Just think, years from now when Amazing Computing grows to ``Byte" status (which began with as humble beginnings as we have), you'll be able to say, ``I was there from the beginning!"

•RC•

INSIDE CLI:

Part One

by

George Musser Jr.

CLI stands not for a new government agency but for **Command Line Interface**, one of the two ways you can work with the Amiga™. In CLI, you type instructions on the keyboard, instead of pulling down menus and dragging icons around the Workbench with the mouse.

Workbench is often easier to use, but CLI unleashes many features of the operating system hidden from Workbench. For instance, you can speed up the machine by storing oft-used programs in the RAM disk, and you can create a list of commands that the Amiga executes automatically whenever you turn it on. For copying and deleting many files, you may find typing faster than pointing the mouse.

To enter CLI, you must create a CLI window. Open the **System** drawer on your Workbench disk, or whatever disk with which you **booted-up**. Inside the drawer should be a CLI icon, a little cube with **1>** on the front face.

If you do not see the CLI icon, close the System window and double-click **Preferences**. In Preferences, there is a switch labelled CLI on the left side of the screen. Click "on" and then "save" in the bottom right corner. The CLI icon should now be inside the System drawer; if it is not, try another copy of your Workbench disk.

When you click the CLI icon twice, a CLI window appears. Now you are in CLI. Since your CLI window rests on top of the Workbench, you can freely alternate between CLI and the Workbench at the click of a mouse button. If you click outside the CLI window, simply click inside the window to resume typing.

CLI understands commands to **list**, **copy**, **delete**, and **run** files, as does Workbench, but instead of using pull-down menus, you type your commands after the prompt, **1>**. This prompt indicates the number of the CLI window. You can open many CLI windows, and each can run a different program or perform a different set of commands. To get a new CLI window, double-click the CLI icon or type **NEWCLI** and press **RETURN**. To get rid of a CLI window, type **ENDCLI**.

Tree Climbing and the Amiga™

The Amiga organizes files into tree-like structure. Each disk has a **main**, or **root**, **directory**, which contains files and **subdirectories**. Each subdirectory can hold more files and subdirectories, and so on. In this way, you can group your files into a hierarchy. For instance, a directory named DOCUMENTS might contain subdirectories called MEMOS and LAB REPORTS.

As with climbing a tree, you can only be in one place at a time. When you open a CLI window, you start at the root directory. The **CD** command moves you around. Type **CD** followed by a **pathname**. The pathname is a set of map directions telling the Amiga where to go. The simplest pathname is the name of a directory. **CD DEVS**, for example, drops you into the DEVS directory. If the name of a directory is two or more words, enclose the pathname in quotes, as in **CD "LAB REPORTS"**.

Pathnames can include subdirectories. **CD DEVS/PRINTERS** carries you to the **PRINTERS** subdirectory of the **DEVS** directory. **CD /** will take you backwards. From **PRINTERS**, **CD /** returns to **DEVS**. Another **CD /** puts you back in the root directory. **CD :** takes you straight to the root directory no matter where you are.

To jump over to another disk, type the diskname followed by a colon, such as **CD EXTRAS:**. A System Request window will ask you to insert the disk, if it is not already in a drive. **CD DF1:** makes whatever disk is in the first external drive the current directory.

Enter **DIR** to see what files are in your current directory. **DIR** lists subdirectories with a "(dir)" after the name, and shows an alphabetized list of files in the directory. To list files in other directories without changing your current directory, type **DIR** followed by a pathname. **DIR DOCUMENTS** lists the files in **DOCUMENTS**, while **DIR DOCUMENTS/MEMOS/1985** looks at the directory 1985. Because the directory **LAB REPORTS** has two words, you must enclose the pathname in quotes, that is, **DIR "DOCUMENTS/LAB REPORTS"**.

If you type **DIR** followed by a filename, the Amiga will search for a file with that name.

The command **LIST** also lists files, but tells you how large they are and when they were last updated. You can **LIST** to search for files with something in common. To get all files ending in ".INFO", use **LIST P=#?.INFO**; the **#?** is called a wildcard and matches with any file name. **LIST SINCE 01-JAN-86** shows all files updated since New Year's Day; **LIST UPTO 01-JAN-86** yields all files last updated before 1986.

Manipulate Your Files

Once you know where the files are, you can manipulate them with a variety of CLI commands. To erase a file, type **DELETE** and the filename. To make a new copy of a file, type **COPY**, the filename, and the name of the copy. **DISKCOPY DFO: TO DF1:** copies a disk in the internal drive to a disk in the external drive, so you do not keep swapping disks.

To run a program, type its name. To run a program and still be able to enter CLI commands, use **RUN** followed by the program name. For instance, **CLOCK** will show the clock, but you must close the clock to continue using CLI. If you type **RUN CLOCK**, you can keep track of time and still execute CLI commands.

The **RENAME** command changes the name of a file; the syntax of the command is **RENAME <old name> <new name>**. You can use **RENAME** to transfer files from one directory to another by including pathnames in front of the file names. For instance, typing the command line **RENAME DOCUMENTS/MEMOS/1985/JAN01 DOCUMENTS/MEMOS/1986/JAN01** moves the file **JAN01** from the directory 1985 to the directory 1986.

You may even **RENAME** the CLI commands. The commands are stored in directory **C** on the Workbench disk. Go to the **C** directory with a command **CD C**. Then input the command, **RENAME RENAME REN**. From now on, type **REN** when you want to rename a file.

CLI conveniently lets you enter as many commands as you like, even while it is executing a command. You can stack as many commands as you wish, and the Amiga will perform them in turn. For example, if you type **COPY JAN01 JAN01.BACKUP**, hit **RETURN**, and then type **COPY JAN02 JAN02.BACKUP** and **RETURN** without waiting for the first **COPY** to finish, when the Amiga

finishes COPYING JAN01, it will display a prompt (1>) and proceed to COPY JAN02.

The CLI and The Ram Disk

If you are using CLI and you remove the Workbench disk, a System Request window will ask you to insert the Workbench disk whenever you want to execute a command. Fortunately, you can avoid endless disk swapping by storing CLI commands in the **RAM disk**.

A RAM disk is an area in memory that a computer treats like a real disk. RAM disks lose their contents if the system crashes or you turn the machine off, but RAM is much faster than a physical disk drive. Never put files on a RAM disk if you can not afford to lose them. Since you have copies of CLI commands on the Workbench disk, you can gain the advantage of speed at no risk. Files on the RAM disk consume memory, so only put files there that you need.

The Amiga treats the RAM disk like another disk drive. The RAM disk is referred to as **RAM:**. You can COPY RENAME RAM:, and the RENAME command will be in the RAM disk. RAM: on the Amiga differs from conventional RAM disks on other computers in that memory is **dynamically allocated**. In other words, RAM: gobbles only as much memory as it needs, and no more. Other RAM disks require you to specify the amount of memory for the disk when you boot-up.

Dynamic allocation saves memory, but be careful. When you tell the Amiga to store a file in RAM:, it puts the file in the first place it finds enough memory. If you delete a file, it opens up a hole in memory that another file can use. However, the hole must be big enough to hold a file. If you delete a small file, chances are another program will not fit there, so you are wasting memory.

If you fill the RAM: and start deleting files here and there, you may open up a lot of holes in memory, none of which are big enough for a useful program. Unfortunately, the memory meter on the Workbench only adds up the size of the holes and does not tell you how big the chunks of memory are. The key to using RAM: is to first empty out memory, and then put those files you need in all at one time.

ASSIGN

Once you copy CLI commands into RAM:, you must tell the Amiga that the commands are in RAM: and not in directory C of the Workbench disk. The **ASSIGN** command does the trick.

The Amiga looks in certain places to find certain types of files. Type ASSIGN to see those places. They are called **logical devices**. C: is the logical device for CLI commands. When you boot-up, C: is mapped to the C directory of the Workbench disk. If you load CLI command into RAM:, you want to map C: to RAM:. To do this, type **ASSIGN C: RAM:**. From then on, all CLI windows will look to RAM: when you type a command.

Instead of loading CLI commands into RAM:, you may just redefine the disk on which the directory C is kept; that is, you may want to insert a disk that has its own directory C. Type **ASSIGN C: DFO:C:**, and the Amiga will look to that disk, instead of the Workbench disk, to find the C: logical device.

Creating your own Startup Sequence

It would be nice if the Amiga automatically copied important CLI commands into RAM: and did an ASSIGN C: RAM:. No problem. Look in directory S of the Workbench disk for a file named **STARTUP-SEQUENCE**. Whenever you boot-up or reset the machine, it looks for STARTUP-SEQUENCE. This file is

called an **executable file**, since it contains a list of CLI commands for the Amiga to execute.

Use **TYPE S/STARTUP-SEQUENCE** to take a look. The **ECHO** commands print messages on the screen. The **LOADWB** command loads in Workbench; the **ENDCLI** command closes down the Amiga DOS window and puts you in Workbench, while the **> nil:** keeps CLI from displaying the message that the CLI window has been closed.

You can stick all sorts of things into **STARTUP-SEQUENCE**. To change the file, type **ED S/STARTUP-SEQUENCE**. **ED** is a simple editor. Once you are in **ED**, you can move around the screen using the arrow keys. To perform commands, hit **ESC** and a letter.

ESC D delete the line where the cursor is located.

ESC I "string" inserts a line with "string" on it before the line where the cursor is.

ESC A "string" inserts the line after the cursor.

ESC J join two lines together.

ESC T goes to the top of the file.

ESC B to the bottom of the file.

ESC Q quits without saving any changes you have made.

ESC X saves changes and puts you back in CLI.

By using **ED**, you are ensured you will have a CLI window every time you boot-up. Since you have removed the line containing **ENDCLI**, the CLI window will remain open. Behind it will lie Workbench, so you will need to shrink the CLI window in order to use Workbench.

To make the Amiga ask you for the date, get back into **ED S/STARTUP-SEQUENCE**. Go to the third line, which says **echo "Use Preferences..."**. Delete the line with an **ESC D**. Then type **DATE ?** and press **RETURN**. Type **ESC X** to save.

Similarly, if you inserted a line with **STACK 8000** and another line with **ABASIC** you would automatically drop into **ABasic™** upon boot-up; use **STACK 4096** and **AMIGABASIC** for **AmigaBASIC™**.

You might also insert lines such as **COPY C:RENAME RAM:** to copy CLI commands into **RAM:**, afterwards, include an **ASSIGN C: RAM:**. To speed up Workbench, insert the following lines:

MAKEDIR RAM:SYSTEM

COPY SYS:SYSTEM/DISKCOPY RAM:SYSTEM

ASSIGN SYS: RAM:

Now, when you want to copy a disk in Workbench, Workbench will not ask you to insert the Workbench disk.

Learning CLI can open up new possibilities for you and your Amiga. Your best bet is to experiment with the commands until you are familiar with them.

George Musser Jr.
P.O. Box 2656
Brown University
Providence, R.I. 02912
CC004049@BROWNVN1.BITNET

•RC•

CLI

Command Summary

A quick guide to the major commands of
the AMIGA™ Command Line Interface

from
Amazing Computing™

Summary of CLI Commands

by
George Musser

ASSIGN <device>: <volume(:directory(/file))>

Define location of logical devices, including S:, L:, C:, DEVS:, FONTS:, LIBS:, and SYS:. AmigaDOS™ looks in these places to find the files it needs.

S: contains executable files.

L: contains programs that check diskettes and handle the **RAM: disk**.

C: contains CLI commands.

DEVS: contains I/O device drivers, printer information files, and user-defined preferences.

FONTS: contains text fonts, named after rare gems.

LIBS: contains system libraries.

SYS: is the disk with which you boot-up.

The logical devices default to the respective directories on the disk with which you booted-up, e.g. Workbench/C.

With **ASSIGN**, you can boot up with one disk, and then start using another disk instead of getting that annoying "Insert Workbench Disk" message. You can also copy the CLI commands to **RAM:** and **ASSIGN C: RAM:**. **ASSIGN** will even let you create your own logical devices. For example, you could create synonyms for CLI commands by assigning a device name to a file, e.g. **ASSIGN D: C: DELETE**.

BREAK <task number> (C) (D) (E) (F) (ALL)

Execute appropriate control code in the specified task. **BREAK** has the same effect as going to that window and typing control-C, control-D, etc. If you do not type a code after **BREAK**, the Amiga assumes control-C, which breaks the task. Control-D stops an executable file after the next command. **BREAK n ALL** sets all the signals.

CD <(volume:directory)>

Change Directory of the current CLI> : is the root directory. **CD /** moves one directory toward the root. **CD volume:** requests that volume.

COPY <directory/file> (TO) <directory/file> (ALL(QUIET))

Copy a file or a directory. To copy all the files in a directory, use **COPY A D ALL**. Use **COPY A D ALL QUIET** if you do not wish to be told which files are being copied. Single-driver users, beware! COPY seems to have almost no memory buffering, so COPYing even a small file may take a lot of swaps. If you swap more than, say, 20 times, think about selecting cancel on the next System Request.

Date (dd-mmm-yy) (hh:mm(:ss)) (TO file)

Set system time and date. Type the 24-hour time and the date in 01-Jan-86 format. You can **DATE 13:00 TODAY**, **DATE 13:00 TOMORROW**, or any day of the week. Days of the week set the date to the next date after the one stored in the machine.

DELETE <directory/file> (<directory/file>...) (ALL QUIET))

Kill one or more files. Once you **DELETE** a file, it is gone for good. You can delete up to nine files on a single **DELETE** line, or even an entire directory using **DELETE <directory> ALL**.

DIR <volume(:directory(file))> (OPT A/D/I)

List filenames in alphabetical order and in two columns.

OPT D will list only directories.

OPT A will list all the files in the subdirectories, also.

OPT I will prompt you after every filename for an option.

q quits the listing

t types out the file (while here, **c** stops the typing and return for prompt)

b goes back up a file

DEL deletes (erases) the file

e looks inside subdirectories.

DISKCOPY DF_n: (TO DF_m:) (<new name>)

Copy a disk, track by track.

ECHO <string>

Print string on console.

ED (FROM) <file> ((SIZE)<n>)

Full-screen editor. The size, greater than or equal to 5000 bytes, is as long as the file is allowed to be.

ESC and **D** delete the line where the cursor is located.

ESC I "string" inserts a line with "string" on it before the line where the cursor is.

ESC A "string" inserts the line after the cursor.

ESC J join two lines together.

ESC T goes to the top of the file.

ESC B to the bottom of the file.

ESC Q quits without saving any changes you have made.

ESC X saves changes and puts you back in CLI.

EDIT

A cryptic, frustrating little line editor.

ENDCLI

Close current CLI window.

EXECUTE <executable file>

Run executable, or batch, file. An exec is a list of CLI commands.

FAULT <n>

Display error message corresponding to error code n, you do not need to look in the back of the manual for the error code .

FILENOTE <file> <string>

Add comment to **INFO** file. You can see the comment when you **LIST** files or use the **INFO** command from Workbench. Note that Workbench 1.0 has a bug that prevents you from commenting a file from Workbench, forcing you to use **FILENOTE**. This bug has been fixed in Workbench 1.1.

FORMAT DRIVE DF_n: NAME <string>

Disk format and verify. The disk doesn't contain DOS, so use **INSTALL** to create a disk with which you can boot-up. You must **FORMAT** a disk before trying to store programs on it.

INFO

Check information on available disks. **INFO** tells you how full the disks are, whether the write-protect tabs are on or off, and the disk names. It also tells you how full **RAM** is.

INSTALL DFn:

Put the system on a disk. Equivalent to **SYS** in **MSDOS**.

JOIN <file> (<file>...) AS <new file>

Combine up to 15 files into one.

LIST <directory> (P=<pattern>) (S<pattern>) (KEYS) ((NO)DATES) (TO<file>) (SINCE<date>) (UPTO <date>) (QUICK)

List file names, sizes, protect flags, date and time of last modification. Use the **QUICK** option to list just filenames. Use **P=*?.INFO** to list all >INFO files; ***?** is the wildcard.

LOADWB

Invoke Workbench. The Workbench will appear behind the CLI window from which **LOADWB** was typed.

MAKEDIR <directory>

Create Directory.

NEWCLI

Open a new CLI window. Tasks in this window can run simultaneously with tasks in other windows.

PROMPT <string>

Redefine CLI prompt to a desired string. Use **%n** for the task number.

PROTECT <directory/file> (R) (W) (E) (D)

Set the protect flags for file.

R permits reading.

W permits writing.

E permits execution.

D permits deletion.

PROTECT <file> RWE would turn on the R, W, and E flags and turn off the D flag. This would prevent deletion of the file.

RELABEL DFn: <name>

Rename a disk.

RENAME <directory/file> (TO/AS) <new name>

Rename directory or file. Will also take a file into a new directory.

RUN <program> (<arguments>)

Execute program and offer a new CLI prompt. If you execute a program by typing its name, CLI will wait for the program to end before prompting.

SEARCH <directory/pattern> (SEARCH) <string> (ALL)

Scan ASCII file(s) for the specified string.

SORT <file> <new file> (COLSTART <n>)

Sort text file alphanumerically. COLSTART is the column in which the sort keys begin.

STACK <n>

Set stack size. Defaults to 4000 bytes. You need 8000 for ABASIC™, 4096 for AmigaBASIC™.

STATUS (<n>) (FULL) (TCB) (SEGS) (CLI/ALL)

Determine the name and condition of the CLI tasks. If you type STATUS by itself or with CLI or ALL, AmigaDOS™ tells you the names of the tasks currently running through CLI windows.

TYPE <file> (OPT N/H) (TO <file>)

Dump file contents to the console or another file. You can use type on non-text files, but it usually sets the font to gibberish, so you need to type control-O to reset. Use OPT H for hexadecimal code; OPT N to include line numbers.

WAIT <n> SEC/MIN || UNTIL hh:mm

Pause for x seconds, y minutes, or until a given time.

WHY

Ask the Amiga why the last command did not work. It generally does not know, though.

Exec file commands

The following group of commands make sense only within an **exec**, or **executable** file.

FAILAT <return code>

Instruct **exec** to end if the error equals or exceeds argument.

IF (NOT) (WARN) (ERROR) (FAIL) (<string1> EQ <string2>) (EXISTS<file>)

ELSE

ENDIF

Conditional execution of a group of lines. Put the keywords **IF**, **ELSE**, **ENDIF** on lines by themselves. **ELSE** is optional.

LAB <label>

Label for **SKIP** command in **exec**.

SKIP <label>

Goto <label>, as defined by **LAB**. You can only **SKIP** ahead.

QUIT

Exit **exec**.

•RC•

Don't Be Shy! Communicate from your

AMIGA

With EITerm,

you can open your Amiga up to the world of telecommunications. EITerm is a complete package for terminal emulation and file transfer.

EITerm can upload and download files using Kermit and Xmodem protocols.

EITerm will be available first quarter 1986.

Suggested retail price is \$75.00 US

Dealer Inquiries Invited.

Elcom Software

16 Oak St. #2
New Brunswick, NJ 08901

Amiga is a trademark of Commodore-Amiga Inc.
EITerm is a trademark of Elcom Software.

Hey Amiga Developers!

Need a database capability in your application?

How about:

- o Arbitrary record and key construction.
- o Multiple indices freely intermixed.
- o Add, modify, or delete indices at any time.
- o Duplicate or non-duplicate key values.
- o Extensive caching of both key and data sets.
- o Multiple data sets concurrently.
- o Partial key searches.
- o Full error checking and reporting.
- o 16Mbyte data set capacity.
- o Full documentation.

AMIGA binaries: \$85.00 (US)

Manual Only: \$25.00 (US)

Sources: \$450.00 (US)

Sources will compile and operate under UNIX Version 6, 7, System III, System V, 4.1BSD, 4.2BSD, 4.3BSD.

Manual cost will be applied to future purchase.

Dealer Inquiries Welcome

Advanced Systems Design Group

280 River Rd. Suite 54A Piscataway, N.J. 08854

AMIGA is a trademark of Commodore-Amiga, Inc.
UNIX is a trademark of AT&T Bell Laboratories, Inc.

New Amiga Products From The Developers of Amiga C.

Amiga C Compiler—\$149.95

Everything you need to develop programs on the Amiga, including a full set of libraries, header files, an object module disassembler, and sample C programs.

Unicalc—\$79.95 A complete spreadsheet package for Amiga, with the powerful features made popular by programs such as VisiCalc, SuperCalc, and Lotus 1-2-3. Unicalc provides many display options and generates printed reports in a variety of formats and print image files. Supports 8192 rows of 256 columns, and includes complete on-line help.

Lattice MacLibrary—\$100.00

The Lattice MacLibrary is a collection of more than sixty C functions enabling you to rapidly convert your Macintosh programs to run on the Amiga. This allows you to quickly and efficiently take advantage of the powerful capabilities of the Amiga.

Lattice Make Utility—\$125.00

Automated product generation utility for Amiga, similar to UNIX Make, LMK rebuilds complex programs with a single command. Specify the relationships of the pieces, and automatically rebuild your system the same way every time.

Text Utilities—\$75.00 Eight software tools for managing text files. *GREP* searches for specified character strings; *DIFF* compares files; *EXTRACT* creates a list of files to be extracted from the current directory; *BUILD* creates new files from a batch list; *WC* displays a character count and a checksum of a specified file; *ED* is a line editor which utilizes output from other Text Utilities; *SPLAT* is a search and replace function; and *FILES* lists, copies, erases or removes files or entire directory structures.

Lattice Screen Editor (LSE)—

\$100.00 Fast, flexible and easy to learn editor designed specifically for programmers. LSE's multi-window environment provides the editor functions such as block moves, pattern searches, and "cut and paste". Plus programmer features such as an error tracking mode and three assembly language input modes.

OTHER AMIGA PRODUCTS AVAILABLE FROM LATTICE:

Panel: Screen Layout Utilities—\$195.00

Cross Compiler:

MS-DOS to Amiga C—\$250.00

dBC III:

library of data base functions—\$150.00

Cross Reference Generator—\$45.00

With Lattice products you get *Lattice Service* including telephone support, notice of new products and enhancements, and a money-back guarantee. Corporate license agreements available.



Phone (312) 858-7950 TWX 910-291-2190

INTERNATIONAL SALES OFFICES:

Benelux: De Vooght. Phone (32)-2-720-91-28. England: Roundhill. Phone (0672) 54675

Japan: Lifeboat Inc. Phone (03) 293-4711 France: SFL. Phone (1) 46-66-11-55

The Amazing..... C Tutorial

part one

by
John Foust

The Amiga has drawn more people to the C language than any other computer. Amiga owners who enjoy programming soon realized that C holds more promise than BASIC.

In this tutorial, C will be compared and contrasted to other languages you may know, such as Pascal, BASIC and FORTRAN. Some parts of C are easily recognized if you know another computer language. For example, C uses a **function** called **printf()** to print messages on the screen. It's name means "**print formatted.**"

If the word "**function**" is new to you, substitute the word "**subroutine**" or "**procedure.**" A function is a part of a program that performs a given task, such as printing messages to the screen, or reading a file from the disk.

C will be presented in bite-sized chunks, with enough background in this first lesson to help you understand the C manuals, and compile and link your first C program. Some background with other computer languages is assumed, but parallel examples will be given in BASIC, FORTRAN, and Pascal.

First, a little history. Most C tutorials include a history for several reasons. In many ways, the history of C is an oral tradition passed from programmer to programmer. These details are like the handsigns of a secret club. They tell others that you have an insider's understanding of the language. A study of C's family tree also explains the historical forces that guided the evolution of this language.

Brian Kernighan and Dennis Ritchie wrote the book "**The C Programming Language**" in 1977. C was first developed under the Unix™ operating system in the early 70s. In fact, the present-day Unix operating system is written in C. C retains several conventions from the Unix system, such as the way standard C programs accept input and send output.

In our case, there is another reason for the history lesson. C and the Amiga are close kin. AmigaDOS, the Amiga operating system, is based on the TRIPOS operating system. A key developer of TRIPOS, Martin Richards at Cambridge University, also wrote a language called BCPL, which influenced the design of the C language.

Also, Commodore-Amiga used C to write Intuition and Workbench, the windowing software on the Amiga.

C has become the preferred language for software developers. C is fast, nearly as fast as hand-written machine language. C will let you access the deepest reaches of your Amiga in a way that BASIC never can. C is powerful. It is also intimidating to the casual observer. Be prepared to be confused!

C is very flexible. It lets programmers do what programmers prefer to do almost anything. While Pascal may refuse to allow the assignment of a floating

point variable to a character variable, and BASIC refuses to assign a string variable to an integer variable, C will gladly perform the assignment. Chances are, the resulting mixture of data is nearly useless, but C assumes you know what you are doing, and does exactly what you asked.

This language characteristic is called "**typing**". Pascal is a strongly typed language, since Pascal's assignment operator '=' doesn't let you assign a floating point number to a character variable. BASIC is less strongly typed, since '=' will assign floating point numbers to integers, and vice versa.

However, C is weakly typed. It might give a warning when you put a square peg in a round hole, but the program will do it just the same.

(If you feel lost already, floating point numbers are numbers with a fractional component. Computers must represent 3.12 as a floating point number, while 1003 can be stored as an integer. Each type requires a different amount of memory space for storage.)

While some advanced programmers regard weak typing as flexibility, the novice, accustomed to Pascal and BASIC, might well regard it as a curse. C has few warnings of the kind you might get from a BASIC interpreter or Pascal compiler.

C is a **compiled language**, as is Pascal. If you know Pascal, C will come easily. "**Compiled**" means that a program is converted to another form before execution. FORTRAN is also compiled. For these languages, a programmer enters a program's source text with an editor, and then **compiles** the source code with the compiler, producing an **object module**. Object modules are **linked** with a linker program to produce an **executable program**.

The compiler translates the source code to this computer's machine language, and makes a list of variables and functions used within this program.

For example, a program might use the printf() function. This function may be part of the C language, but it is defined somewhere else than this source code.

The printf() function is said to be **unresolved** or **external**, since the compiler can't generate machine language for it. An object module is composed of this mix of machine language and the list of unresolved references.

After the program is compiled, all references to functions and variables must be resolved with the linker. If you buy a C compiler, you also get several collections of pre-written and pre-compiled functions, called **libraries**.

Standard C functions such as printf() are included in one library, while sin(), cos() and the rest of the extended math functions are in another.

The object module produced above is linked with one or more libraries. The linker uses the object module's list of unresolved functions and variables to search each library.

If a function is found in a library, its machine language and its own list of unresolved references is searched, until all references are found. Only then can the linker produce an executable program. This file is composed of only the functions needed by this program. The other library functions are left out.

If the linker can't find a reference, it cannot produce a correct program, since some of the program is

missing. The linker program will stop after printing a list of **unresolved externals**, or functions and variables it couldn't find in the libraries.

In this case, the programmer would examine the linker's output, and decide on a correction. They might check the source text for spelling errors. If `printf()` was spelled `prnitf()`, the linker would never find the function `prnitf()`, even though it knows about `printf()`.

The programmer might have forgotten to include the library of math functions, if `sin()` and `cos()` appeared in the linker's output list. A programmer's ability to correctly interpret linker output grows with time.

This process of compiling and linking means a lot of typing in the CLI. If you have purchased Lattice C for the Amiga, the compiler disk has an **EXECUTE** file called **'make'**. If you have entered a C program called **'SAMPLE.C'**, you can compile it by typing **'EXECUTE MAKE SAMPLE'**. This EXECUTE file contains all the CLI commands necessary for compiling and linking a C program, saving you minutes of detailed typing.

Let's look at a simple program in C, BASIC, and Pascal.

It will multiply two numbers and print the result:

In C:

```
#include "stdio.h"
```

```
/* a program to multiply two numbers */
```

```
main()
```

```
{  
  int a,b,c;
```

```
    a = 3;  
    b = 4;  
    c = a * b;  
    printf( "The answer is %d\n", c );  
}
```

In BASIC:

```
5 REM a program to multiply two numbers  
10 a = 3  
20 b = 4  
30 c = a * b  
40 print "The answer is "; c
```

In Pascal:

```
program add(input,output);
```

```
integer a,b,c;
```

```
(* a program to multiply two numbers *)
```

```
begin
```

```
  a := 3;
```

```
  b := 4;
```

```
  c := a * b;
```

```
  writeln( 'The answer is ', c );
```

```
end.
```

First, you might notice that a C program has a lot of punctuation. C uses punctuation in the same way Pascal uses 'begin' and 'end'. This makes C programs harder to read than Pascal, for instance. While Pascal uses 'begin' and 'end' to mark the start and end of procedures and functions, C uses the curly brackets '{' and '}'.

Also, Pascal and C separate program comments from program code with punctuation. C uses pairs of `/*` and `*/`, while Pascal uses `(*` and `*)`. BASIC uses the REM statement at the beginning of the line to mark a comment.

Both Pascal and C are compiled languages. Notice that the Pascal and C examples explicitly declare the variables `a`, `b` and `c`. Pascal says 'integer,' while C uses 'int' to declare an integer variable.

This is common for compiled languages. A compiler needs to know the **type** of a variable before it is used in the program. It will use this to perform the correct

operation, such as addition, when this variable is used later on, since floating point and integer numbers are added differently.

The BASIC example doesn't need explicit type declarations for each variable. BASIC is an interpreted language. When you type RUN, the computer analyzes the text of each line, in turn, to execute the steps of the program.

If BASIC encounters a variable that it doesn't recognize, it assumes that variable is a floating point number. In our BASIC example, a,b and c are floating point variables.

C has other variable types, besides 'int'. It has 'float' for floating point, and 'char' for character variables. Each type has a valid range of values. For example, 'int' is restricted to plus or minus two billion on the Amiga, while 'char' is limited to hold values between +127 and -128.

Integral types, like 'char' and 'int', can be modified by the word '**unsigned**', creating the types '**unsigned char**' and '**unsigned int**'. These have ranges of 0 to 255 and 0 to over 4.2 billion, respectively. If you know your bits and bytes, you will recognize these values as the largest numbers that can be represented in 8 and 32 bits.

Each type has several valid operators, such as addition and subtraction, multiplication and division. As opposed to BASIC, Pascal and FORTRAN, C is flexible about which types can be operated upon. For example, you can add 'char' variables and 'int' variables. The value of a 'char' is the ASCII value of the character stored within it. In BASIC, you would get this value using the ASC() function. In Pascal, ORD() would be

used.

You might notice that the C and Pascal program lines end with semicolon. This is another characteristic of a compiled language. The ';' says to the compiler "the programmer thinks this line is complete." In fact, C program lines can be spread out over several physical lines:

```
printf
(
    "The answer is %d"
    ,
    c
)
;
```

with no ill effect. The compiler **parses** the line in the same way. This isn't possible with most BASIC interpreters.

The line '#include "stdio.h"' is a command for a special part of the compiler called the **preprocessor**. This line means "at this point, also compile the program file named stdio.h." This file contains the extra declarations of variables and functions used by printf() and other "standard I/O."

The preprocessor has many other benefits for C programs; these will be described in the next tutorial.

Look at the line in Figure 1 that uses the function printf(). Printf() has two **arguments**, a character string containing the text "The answer is %d\n" and the name of an integer variable, c. A C programmer might call these the "arguments passed to the function," "the function's arguments," or "the values passed to the function." Arguments are listed between the parentheses, separated by commas.

The first argument passed to `printf()` describes the format of the text we want to see on the screen. It contains two parts that may be unfamiliar, both near the end of the quoted text. (If you know FORTRAN, you might remember the FORMAT command.) The `%d` means "print an integer value here."

The `\n` means "print a newline character," which would move the cursor to a new line, just as when you press RETURN.

If we wanted to print two values using `printf()`, we might say:

```
printf( "We have %d bananas and %d  
apples.\n", apples, bananas );
```

to get the text:

We have 3 bananas and 5 apples.

if the variables 'bananas' and 'apples' contained 3 and 5, respectively.

We could ask `printf()` to print a value in a field of a certain width, as FORTRAN does. If you use `%4d` to format an integer, the number will be printed with enough leading spaces to fill out four character positions.

The `printf()` function has other formatting commands, such as `%x` to print values in hexadecimal. To print a value as an ASCII character, in the same way as the BASIC command `CHR$()`, `printf()` uses a `%c` in the formatting string.

Functions are used extensively in C. While the C language has many standard, pre-written functions

such as `printf()`, your programs will be composed of functions you write yourself.

If you have used Pascal, you are familiar with procedures. Pascal has functions, too, but C functions resemble Pascal functions, since all C functions return a value. C functions cannot nest in the way that Pascal procedures and functions can nest.

If you haven't used Pascal, think of the BASIC functions `SIN()` and `COS()`. These functions return a floating point value based on their argument, the number between the parentheses. All C functions return a value of a given type.

Here is a C program fragment that declares a function that returns an integer, the product of two integer arguments:

```
/* a function to multiply two numbers */
```

```
int mult(a,b)  
int a,b;  
{  
  int c; /* a local variable */  
  
  c = a * b;  
  return(c);  
}
```

The first line declares a new function called `mult()`, that has two arguments, `a` and `b`, both integers. A function declaration can be broken into several parts. First, the function type. Here, we used 'int'. If no type is given, C assumes the new function is of type 'int'.

Second, the function name, 'mult'. All function declarations have a set of parentheses, whether or not the function has arguments. This function has two arguments. The type of each argument listed between the parentheses must be declared before the open

bracket. The open bracket marks the beginning of the code for this function.

The line after the open curly bracket declares a `_local variable_` called `c`. If you are familiar with Pascal, local variables in C work the same as Pascal, keeping in mind the restriction that functions cannot nest in C.

If you aren't familiar with Pascal, a local variable is a variable that can't be accessed by any other function or procedure in a program, even if another function declares a local variable of the same name.

Local variables cannot be accessed by other functions in this program. The value of a local variable is only known while this function is executing. After the function finishes, the local variables disappear, and the storage they use is reused.

The `'c = a * b'` should be clear in any language. The line `'return(c);'` marks the finish of the `mult()` function. The value `'c'` is sent back to the function that asked to use the `mult()` function.

Figure 2 is a complete C program that uses the `mult()` function described above. In this program, the `main()` function calls `mult()` with the values stored in `'r'` and `'s'`.

Since `'r'` holds 3, and `'s'` holds 4, the `mult()` function will return the value 12. Within the `main()` function, the value 12 will be stored in the `main()` global variable called `'t'`. (The difference between **global** and **local** will be shown in a moment.)

A C programmer might say that this `mult()` function was "called" by the `main()` function and the value 12 was "returned" to `main()`. Since C is based on

functions and function calls, programmers often use these phrases.

The words "call" and "return" hint at the underlying structure of C. Both words are used in the same context in machine language programs. Many C instructions can be directly translated to machine language.

In the example in Figure 2, the declaration of the `main()` function has no arguments, and no explicitly declared type, so `main()` returns an `'int'`. `Main()` is a special function. When the C program is started by the operating system, `main()` is the first function to begin execution. Other than that, `main()` is no different than other C functions.

In Figure 2, we declare our `main()` function and our new `mult()` function, which work together to make a simple C program.

(Figure 2)

```
#include "stdio.h"

/* a function to multiply two numbers */

int mult(a,b)
int a,b;
{
    int c; /* a local variable */

    c = a * b;
    return(c);
}

/* All variables declared outside of function
   declarations are global variables, which
   can be used by all functions */

int t; /* a global variable */
```

```
main()
```

```
{  
  int r,s;/* local variables */  
  
  r = 3;  
  s = 4;  
  t = mult(r,s);  
  printf("The answer is %d\n", t);  
}
```

The opposite of **local** is **global**. Global variables are any variables declared **outside** of all the functions in the program. Global variables are declared in the same way as local variables. The syntax is the same: give the type first (such as 'int'), then the variable names (separated by commas, if there is more than one), then the terminating ';'.

Global variables can be read and changed by any function in the program. Most programs have a mixture of local and global variables. The local variables are used for intermediate calculations, since they disappear after the function finishes. Global variables are used for more permanent data, and for data that several functions might need to use.

If a function declares a local variable with the same name as a global variable, the local variable takes precedence.

In the next tutorial, you will learn to use the preprocessor, and declare other types of variables. Several more standard C input and output functions will be demonstrated, along with the special Amiga functions that create simple windows. •RC•

Amiga™ Public Domain Software

from

**Amazing Computing™
&
AMICUS™**

Amazing Computing™
will collect and distribute
Public Domain Software
for the benefit of the Amiga user
community.

You are encourage to give copies of all
public domain software to fellow Amiga™
owners and User Groups

**Public Domain Disk #1 available
now!**

**Public Domain Disk #2 available
February 1986**

Send \$7.00 (\$6.00 for Amazing Computing
Subscribers) check or money order for each
disk requested

To:
**Public Domain Software
Amazing Computing™
PiM Publications, Inc.
P.O. Box 869
Fall River, MA. 02722**

AMIGAForum

....PCS-61

by Bela Lubkin
aka CIS#76703,3015

When you need information, help may be as close as your modem and a CompuServe™ connection. PCS-61 or the AMIGAForum on CompuServe™, is a special electronic bulletin board filled with Amiga owners working through the inner mysteries of their Amigas and software.

We are fortunate to have a systems operator (SYSOP) from the AMIGAForum here to explain the tricks and techniques of the Forum as well as the benefits. Bela Lubkin is an independent consultant and avid Amiga user. It is rumored, he is "constantly" on CompuServe™, so who better to guide us into the wide world of electronic communication than Mr. Lubkin. --Editor--

CompuServe's Amiga Forum opened on December 2, 1985. During the previous weeks, the three volunteer Sysops (Systems Operators), George Jones, Lee Savory and myself, worked long hours to arrange and establish the forum. George and Lee are CompuServe employees, George in Compiler Support and Lee in Network Software Development. I am an independent consultant in California.

We volunteered our services as Sysops and founded the Amigaforum to encourage the widest possible distribution of information, ideas and programs relating to the Amiga.

Joining CompuServe

CompuServe Information Service™ (CIS)™ is the public branch of CompuServe, Inc.'s online information service. To join the service, you need to purchase a CIS Starter Pack™. These kits are sold with a variety of communications packages. If one did not come with your communications package, ask your local dealer or

contact:

CompuServe Inc.

5000 Arlington Centre Blvd.

Post Office Box 20212

Columbus, Ohio 43220

(800) 848-8990

(in Ohio, (614) 457-8650)

Once you have a Starter Pack, set your equipment to the parameters listed in the pack. Your local dealer should be helpful if you are unsure of the requirements. Once your equipment is ready, dial your local **node**, telephone connection, (telephone numbers are listed in the Pack) and follow the interactive instructions.

Once a CIS member, you are free to roam about the service. To join the AmigaForum type **GO AMIGAForum** or **PCS-61** at any ! prompt. (When I give a CIS command, I will list its shortest abbreviation in UPPERCASE, with the optional parts in lowercase. Thus GO AMIGAForum can be abbreviated to G AMIGAF).

You will see a menu, one of whose choices is to join the Forum. Choose this option, and in response to the next question, type your full name as you wish it to appear in any messages that you send. The system will print the Short Bulletin and then give you the main Forum Function Menu:

The Amiga Forum

FUNCTIONS

- 1 (L) Leave a Message
- 2 (R) Read Messages
- 3 (CO) Conference Mode
- 4 (DL) Data Libraries
- 5 (B) Bulletins
- 6 (V) View Member Directory
- 7 (SS) Set Subtopic
- 8 (OP) Set User Options
- 9 (H) Help
- 10 (E) Exit from The Amiga Forum

Enter choice :

Chances are, you will quickly find the menus too bulky and repetitive. As soon as this happens, you can disable the menus with the **Options** command. This gives you a menu which includes the **MENU** option. Type **MENU No** and the menus are gone! If you become lost, return to the **Options** command and replace the menus.

When you log into the forum you will find five subtopics currently available:

SUBTOPICS

- 0 Forum Business
- 1 Software
- 2 Hardware
- 3 Telecommunications
- 9 Software Developers
- A All Authorized Subtopics

The organization is meant to be skeletal; more subtopics will be opened as the need becomes apparent. (For example, it seems likely an Entertainment topic will be added).

You have joined CompuServe and the Amiga Forum. Now what can you do? As in all CompuServe Forums, the Amiga Forum offers three main features:

The Message Base

Data Libraries

Conference facility.

The Message Base

First, explore the message base. The message base is a fast paced, ongoing discussion of current events. You will want to read a few messages before attempting to leave one. Use the **Read** command.

There are several ways to **Read** messages; for now, let's use the **Thread** method: type **T**. The next prompt asks you where to start. Type **N** for messages that are **New** to you.

The oldest message in the Forum will be displayed, followed by a menu of disposition options. Pressing Return prints the first reply (if any). Each successive Return prints the next reply, until the 'thread' has been exhausted. Then the next original message prints. This continues until the entire message base has been covered. If a thread does not interest you, type **SKIP ALL** to ignore it.

If you want to reply to a message, type **REply**. The message editor is called. Type your message, being careful to limit each line to 79 columns. When your message is printed, CIS will wrap the text to the width of the reader's terminal. To prevent a particular line from being appended to the previous line, either indent

it, or start it with a period ('.') in the first column. When you are finished, enter a blank line.

Now **Preview** your message to be certain it is correct. If there is anything wrong, use the **Edit** and **List** commands to fix the problem. You can receive more information on these commands by entering a question mark ('?') followed by the command in question:

'? **Edit**' (the space is necessary). As a matter of fact, ? will work at almost every prompt in the Forum.

When all is well, enter **Save** to send the message.

To start a new thread, use the **Leave** command from the main Forum prompt. Anyone can leave a message in one of the 5 predefined subtopics, and expect some sort of reply within days.

In fact, the message base holds only 384 messages; old ones scroll off into the 'bit bucket' in favor of new ones. Since about 100 new messages are posted daily, it is necessary to return within about four days if you want to receive any replies.

As the Forum becomes more active, this time will grow shorter. It is wise to note the age of the oldest current message when you first join the Forum and use it as a guide of message survival time.

If you are starting a new set of messages, it is very important that you use a title that describes the content of your message. This title becomes the Thread that will string all your future replies.

If you want pricing information on the optional 5.25" disk drive, use a title such as 'Price of 5.25" drive?' rather than 'Disk drive'. This helps anyone using **QS**

(Quick Scan) to choose which threads to read.

You would also want to store the above message in subtopic 2, Hardware. (More documentation on QS can be found online by typing **HELP QS**).

Data Libraries

Associated with each message subtopic is a **Data Library**. The **DLs** store files of all types. To enter a DL, just type **DLn**, where n is the number of the desired topic.

For instance, type **DLO** to visit the Forum Business library. Help files are stored here, so it is a good place to start. Type **BROWse** to look through the data library files. A list of keywords and a description is printed for each file.

You may choose to read or download any file. (Downloading is transmitting the file from CIS to your computer, while uploading is sending a file from the computer to CIS).

In the **BROWse** mode, you are offered options after each description of a file. You may pass to the Next file or choose a downloading method.

Reading lists the text of the file to your terminal.

Downloading requires some type of protocol. CIS supports **XMODEM** as well as their proprietary '**A**' and '**B**', protocols. Note that an especially tolerant XMODEM is required to prevent network delays from aborting the transfer. Most modern communications programs handle this correctly.

Downloading is actually quite a complicated process. You will have to deal with protocols, ASCII text vs

binary files, adding or removing line feeds, and several other topics. But don't worry about it: tools are available in the DLs, and knowledgeable help is only a message away.

ASCII text files can be **Read** and captured by your communications program, while binary files (including compiled, executable programs) require protocol downloads. If your attempts to use XMODEM are unsuccessful, leave a message to the Sysops and they will help you.

Uploading to the Data Libraries

When you are familiar with the system, you may wish to upload your own creations and files. Before you do, however, read the **DL bulletins**. The bulletins can be accessed with the **B** command at the main Forum prompt. The DL bulletin gives suggestions about file naming conventions and other important considerations.

CIS filenames are limited to a 6 character name and a 3 character extension, as in AVAIL.TXT, a file of available software and hardware, maintained by member Richard Rae, (yes, we used only 5 letters in the first section, just remember, no more than 6). You will want to plan ahead, determining a proper file name as well as appropriate keywords and a description.


The keywords help people search the DLs for a particular type of file. If you are going to upload a 'Pong' game written in AmigaBasic, some appropriate keywords might be PONG, GAME, AMIGABASIC, MSBASIC, GRAPHICS and SOUND.

The description should be two or three sentences describing the program, such as:

'This is an exciting implementation of Pong using the 320x200 16 color mode. Requires AmigaBasic. 11K of source, by Bela Lubkin. Please report any problems or suggestions to user ID 76703,3015'.

In Conference

The third area of the Forum is the conference facility. By typing **COnference** at the main Forum prompt, you enter a real time online conference. Any number of people can type single line message to each other, holding an online conversation. Anything you type will be sent to all the other **CO** participants in that channel as soon as you hit Return.

Haven't  You Set
Your **AMIGA'S**
Time **And Date**
Once ☐ Too
Often?

Introducing

A - T I M E

A clock/calendar card with battery-back-up, so you will never have to set the time and date in your **AMIGA**, EVER AGAIN!

PRICE \$49⁹⁵ AVAILABLE: NOW

Coming Soon:

THE AKRON RAM CARD

Expand your AMIGA to its ultimate potential

AKRON TERM + BBS

The standard in terminal emulation. Supports unlimited communications and transfer protocols.

AKRON SYSTEMS DEVELOPMENT (ASD)

P. O. BOX 6408

(409) 833-2686

BEAUMONT, TX 77705

include \$3.50 for shipping and handling
AMIGA is a trademark of Commodore - Amiga inc.

UBZ FORTH™

for the Amiga™

- *FORTH-83 compatible
- *32 bit stack
- *Multi-tasking
- *Separate headers
- *Full screen editor
- *Assembler
- *Amiga DOS support
- *Intuition support
- *ROM kernel support
- *Graphics and sound support
- *Complete documentation
- *Assembler source code included
- *Monthly newsletter

\$85

Shipping included
in continental U.S.
(Ga. residents add sales tax)

UBZ *Software*

(404)-948-4654

(call anytime)

or send check or money order to:

UBZ *Software*

395 St. Albans Court
Mableton, Ga 30059

**Try FORTH
Be Impressed**

*Amiga is a trademark for
Commodore Computer. UBZ FORTH
is a trademark for UBZ Software.

There are 30 channels in each Forum; in general, channel 30 is used for everything and the others are unused -- but there's no reason not to use them. Commands to **CO** must be prefixed with a slash ('/'). The most important commands are **/HELP**, **/EXit**, **/TUNe** for changing channels, **/HANDle** for setting your name, and **/UST** (user status) for seeing who else is on. Unlike elsewhere in the Forum, handles are allowed in CO.

As this is in writing, a formal conference schedule is being drawn up. If it is adopted, there will be two weekly conferences: a Developer's round table at 10pm Eastern/7pm Pacific every Saturday, and at the same time each Sunday, an open conversation for all members.

Of course everyone will be welcome to the Developers' meeting, but the discussion is limited to development issues.

The conference schedule is displayed the first time you enter CO, and is also available from the Bulletins menu (B from the main Forum prompt).

After you've visited the message base, the data libraries and the conference facility, you'll have a pretty good feel for the Forum. When you're satisfied, use the OFF command to sign off of CompuServe (or use GO to jump to another Forum). You will have become one of over 1000 members in one of the fastest growing, most exciting Forums on CompuServe.

See you there! Sysop/Bela Lubkin.

•AC•

Commodore Amiga Development Program

Amiga owners have an excellent path into software and hardware development. While some major computer companies have little or no third party support for development on their equipment, and other companies make the process of attaining development status as difficult as passing a bar exam, Commodore is straight forward in their developer program and support. If you have a credible concept and you either have the skills required to make your concept a marketable reality or have access to those skills, Commodore will accept you as a Developer.

You should consider asking yourself the following:

1. Do you have a company?
2. Are you now marketing products?
3. How would you market your product?
4. What equipment do you now have and utilize?
5. What products are you considering?
6. What will be your estimated release date?
7. What area does your product fall into?
 - A. Entertainment
 - B. Education
 - C. Productivity (business)
 - D. Tools or Languages (graphics)

Your concentration should be on self evaluation. How well do your talents and experiences fit the business you have planned? Will you make up any short comings with friends or relatives? Can you afford to market the product?

To apply, prepare a written proposal stating clearly your intentions and experiences. Give a complete background of your company and suggest why this venture will be a success. When you are certain that your proposal is as perfect as your software will be, mail (please do not call) to:

David Street
Amiga Software Development
Commodore Business Machines
1200 Wilson Drive
West Chester, PA 19380

If you are accepted, what can you expect? The answer is constantly changing as the program is under continual reevaluation.

At this time, you will have an opportunity to purchase a developer's software kit, utilize the technical support staff directly at Commodore, and become a member of a select group on the QuantamLink™ network.

The Software kit contains Lattice C, an Assembler, and the full documentation (hardware -no schematics- and software). At this writing, that includes the cross development software in IBM PC format.

Support telephone lines are available. However, you will probably need to leave your question and receive your answer later. The people are friendly, but do not always have access to all the information when you call.

You will be able to sign onto Quantam's QuantamLink and a forum for discussing Amiga development problems with Commodore Amiga hardware and software technicians. This direct link will provide access to more complete reference materials and technical data.

The forum will provide greater flexibility and allow instant communication between all parties. This environment will increase the amount of development for the Amiga and thus, more software.

•AC•

Amiga Products

Software

Languages & Tools

UBZ Forth™	UBZ Software	\$85.00	Jan '86
Turbo Pascal™	Borland International	TBA	TBA
MultiForth™	Creative Solutions Inc.	TBA	1st Qtr '86
Amiga C Compiler™	Lattice	\$149.95	NOW
Lattice MacLibrary™	Lattice	\$100.00	NOW
Lattice Make Utility™	Lattice	\$125.00	NOW
Text Utilities™	Lattice	\$149.95	NOW
Lattice Screen Editor™	Lattice	\$100.00	NOW
Panel (screen Layout utilities)™	Lattice	\$195.00	FEB '86
Cross Compiler (MS-Dos to Amiga)™	Lattice	\$250.00	NOW
dBCIII (Library of DB functions)™	Lattice	\$150.00	NOW
Cross Reference Generator™	Lattice	\$45.00	NOW
MCC Pascal™	Metacomco	\$99.95	NOW
Cambridge Lisp™	Metacomco	\$199.95	NOW
Marauder (Disk utility)™	Discovery Software	TBA	1st Qtr. '86

Graphics

Aegis Animator™ with Aegis Images™	Aegis	\$139.95	Jan '86
Aegis Images™	Aegis	\$79.95	Jan '86
Amiga Draw™	Aegis	\$199.00	Feb '86
DeluxePaint™	Electronic Arts, Inc.	\$79.95	NOW
Sound Vision™	Hayden Software	TBA	FEB '86

Entertainment

Mindshadow™	Activision	\$44.95	NOW
Borrowed Time™	Activision	\$44.95	NOW
Hacker	Activision	\$44.95	NOW
Julius Erving, Larry Bird, One on One™	Electronic Arts, Inc.	\$35.95	NOW
The Seven Cities of Gold™	Electronic Arts, Inc.	\$39.95	NOW
Arcon™	Electronic Arts, Inc.	\$39.95	NOW
Skyfox™	Electronic Arts, Inc.	\$39.95	NOW
Adventure Construction Set™	Electronic Arts, Inc.	\$49.95	
Articfox™	Electronic Arts, Inc.	\$39.95	NOW
Sargon III™	Hayden Software	\$49.95	FEB '86
Zork I™	Infocom, Inc.	\$39.95	Now
Zork II™	Infocom, Inc.	\$49.95	TBA
Zork III™	Infocom, Inc.	\$49.95	TBA
Wishbringer™	Infocom, Inc.	\$39.95	NOW
Hitchhiker's Guide to the Galaxy™	Infocom, Inc.	\$39.95	NOW
Spellbinder™	Infocom, Inc.	\$49.95	NOW
Sorcerer™	Infocom, Inc.	\$39.95	NOW
Keyboard Cadet™	Mindscape	\$39.95	NOW
Racter™	Mindscape	\$44.95	JAN '86
Halley Project™	Mindscape	\$44.95	FEB '86
Deja Vu™?	Mindscape	\$49.95	2nd Qtr '86

Business

Financial Cookbook™	Electronic Arts, Inc.	\$79.95	NOW
Unicalc™	Lattice	\$79.95	NOW
Maxiplan	Maxicorp	\$125.00	Feb '86
Analyze	Micro-Systems Software, Inc.	\$99.00	Jan '86
VIP	VIP Technologies	\$149.95	Jan '86

Communications

ElTerm™	Elcom Software	\$75.00	1st Qtr. '86
Maxicom™	Electronic Arts/Maxicorp	\$49.95	NOW
OnLine	Micro-Systems Software, Inc.	\$69.00	NOW
BBS-PC™	Micro-Systems Software, Inc.	\$99.00	JAN '86

Hardware

A-Time™	Akron Systems Development	\$49.95	NOW
2Meg-8Meg Card	Akron Systems Development	TBA	TBA
Penmouse+™	Kurta	\$375	TBA
Series ONE™ 200 PPI	Kurta		
8.5 x 11	Kurta	\$695	TBA
12 x 12	Kurta	\$795	TBA
12 x 17	Kurta	\$895	TBA
Series TWO™ 12 x 12	Kurta		
400 PPI	Kurta	\$835	TBA
1000 PPI	Kurta	\$895	TBA
Series TWO™ 12 x 17	Kurta		
400 PPI	Kurta	\$960	TBA
1000 PPI	Kurta	\$1095	TBA
2400B Modem	Micro-Systems Software, Inc.	\$429.00	JAN '86
T-Connect™	Tecmar	\$450.00	DEC '86
T-Card (256K)™	Tecmar	\$795.00	FEB '86
T-Card (512K)™	Tecmar	\$895.00	FEB '86
T-Card (1Meg)™	Tecmar	\$995.00	FEB '86
T-disk (20Meg)™	Tecmar	\$1495.00	DEC '86
T-modem™	Tecmar	\$695.00	FEB '86
T-tape™	Tecmar	TBA	MAR '86

Books

The Amiga Technical Reference Series			
The Amiga Hardware Reference Manual	Addison-Wesley Publishing Co.	\$24.95	March '86
The Amiga Rom Kernel Manual:			
Library and Devices	Addison-Wesley Publishing Co.	\$29.95	March '86
Exec	Addison-Wesley Publishing Co.	\$19.95	April '86
The Amiga Intuition Reference Manual	Addison-Wesley Publishing Co.	\$24.95	March '86
The AmigaDos™ User's Manual	Bantam Books	\$24.95	March '86

The Amazing LISP

TUTORIAL: Part 1

by

Daniel Zigmond

This is the first in a series of Lisp tutorial articles I am writing for Amazing Computing™. Before we have completed all sections, I hope to have covered all of Lisp's fundamental features and some features unique to Lisp on the Amiga™.

WHY Learn Lisp?

Lisp was one of the first computer languages. Its origins are in some rather obscure applications of computer science and, as a result, Lisp is often written off as a language that is only useful for things such as artificial intelligence research. This is not true at all.

Lisp has evolved over the past few decades to become, arguably, the most flexible and powerful programming language in existence. It is used to perform all types of programming. Games, compilers, algebra problem solvers, word processors, graphics packages, tutors and even operating systems are only a few examples of Lisp's applications.

Furthermore, Lisp is very easy to learn. Its syntax is simple and does not require the programmer to memorize lots of cryptic codes. Once one has used Lisp for a short time, Lisp code becomes quite readable and easy to debug.

Cambridge Lisp

Metacomco has released a Lisp implementation for the Amiga™ called **Cambridge Lisp 68000™**. Cambridge Lisp™ is an extremely powerful

programming environment. It provides some highly advanced features almost unheard of in the personal computer world. With some effort, Cambridge Lisp™ can also be made to do the graphics and sound that have made the Amiga™ famous.

The Lisp Interpreter

In Lisp, there are no programs like those in BASIC or Pascal. The programmer simply types what is called a **symbolic expression** and Lisp responds with the **value** of that expression. This makes Lisp an "interactive" language.

Programming in Lisp is no more than having a conversation with the Lisp interpreter. All you need to do to become a proficient Lisp programmer is learn how to write symbolic-expressions.

The best way to do this is to start a conversation and see the Lisp interpreter in action. You should start by typing **Lisp** to AmigaDOS™. When Lisp loads, you will see a copyright notice and some memory statistics on the screen.

Eventually, the prompt **Input:** will appear. This is the signal that Lisp is ready to listen to you. We'll begin by doing some simple mathematics.

Input: 305

Value: 305

Input: -27

Value: -27

Input: (plus 10 3)

Value: 13

Input: (difference 10 3)

Value: 7

Input: (times 10 3)

Value: 30

Input: (quotient 10 3)

Value: 3

Input: (remainder 10 3)

Value: 1

Notice that Lisp always prefaces its response to our expressions with **Value:**. This makes it much easier to follow our conversation because the distinction between the original expression and its value becomes more clear.

The first two expressions we typed are called **atoms** because they are single units of data that cannot be broken down into a simpler form.

The last four expressions are **lists** of three elements each. The first list, (plus 10 3), is composed of the atoms plus, 10, and 3.

The elements did not have to be atoms; they could have been other expressions. For example, the following expression does more complex calculations by **nesting** two expressions within another expression.

Input: (plus (difference 305 157) (times 1 1))

Value: 149

In all our expressions, there is obviously something special about the first atoms in the lists. They seem to determine which mathematical function Lisp uses to compute the value of our expression. These atoms are called **primitives**. They tell Lisp what to do with the remaining elements of the list, called the **arguments**.

In all our function calls, we have used only two arguments for each primitive. This need not be the case. If we want to multiply four numbers, for example, we can do that by typing the expression (times 305 34 76 9).

Some primitives only use a single argument. The primitive minus negates its argument; (minus 1) returns -1, (minus -9) returns 9, etc.

You know all about the structure of symbolic-expressions so we can leave mathematics and begin to look at some of Lisp's more unique features. Cambridge Lisp™ can do some truly amazing feats with numbers but we will go into these later in this series.

List Processing

The word "Lisp" actually stands for "**LIST Processing**" and the ability to do this is considered by many to be Lisp's most interesting feature. It may be hard for you to imagine at this point how atoms within parentheses could ever be exciting but, you will change your mind before this series ends.

You already know what a list is: any number of symbolic-expressions enclosed in a pair of

parentheses. It is important to note that `()` is also a perfectly valid list, called the null list.

List processing is the construction and manipulation of lists. We can build a new list with the list primitive. It takes any number of arguments, just like the original primitives we used, and builds a list out of them. This is best understood by watching it happen.

```
Input: (list 1 2 3)
Value: (1 2 3)
```

```
Input: (list (plus 1 2) (times 0 (plus 2 8)) 5)
Value: (3 0 5)
```

```
Input: (list 1 (list 4 9))
Value: (1 (4 9))
```

Although this is useful, we obviously want to make lists of things other than numbers. This seems like it would be simple enough but expressions like `(list a b)` cause an error. This is because Lisp does not know how to evaluate the expressions `a` or `b` since they are neither numbers or function calls.

However, this should not matter; we do not want the value of the atom `a` in our list, we want the atom `a` itself. We can tell Lisp not to evaluate an expression by placing a single quote before it. For example:

```
Input: (list 'a 'b)
Value: (a b)
```

```
Input: (list (plus 1 2))
Value: (3)
```

```
Input: (list '(plus 1 2))
Value: ((plus 1 2))
```

```
Input: (list 'times 2 3)
Value: (times 2 3)
```

The last primitives we will use in this article are `car` and `cdr`. These are used to take lists apart and examine their individual elements.

The primitive `car` returns the first element of its argument and the primitive `cdr` returns all the rest. By nesting these two together, we can access any element we want.

```
Input: (car '(a b c))
Value: a
```

```
Input: (cdr '(a b c))
Value: (b c)
```

```
Input (car (cdr '(a b c)))
Value: b
```

```
Input: (car (cdr (cdr '(a b c))))
Value: c
```

It would quickly become tiresome to use Lisp if we had to type `(car (cdr (cdr (cdr ...))))` every time we wanted to see the fourth element of a list. Cambridge Lisp™ provides a useful shorthand for deeply nested `car` and `cdr` combinations. We can use `cadr` to mean `(car (cdr ...))`, `caddr` instead of `(car (cdr (cdr ...)))`, `cadar` for `(car (cdr (car ...)))`, and so on.

This makes list processing much easier.

Although we are far from finished with Lisp, the remaining portion will be covered in future installments. Next month we will go further into list processing and learn how to write new primitives to do special tasks for us.



AMICUS™

AMlga Computer USers

The AMICUS Network™

by
John Foust

Welcome to the **AMICUS Network™**. This regular column will represent the AMICUS Network™, an association of Amiga users, developers, and user groups from around the country, Canada, and even Europe. "Amicus" means "friend" and "friendly" in Latin, but also stands for **AMi**ga **C**omputer **US**ers.

The AMICUS Network™ is dedicated to collecting and distributing public domain software for the Amiga. AMICUS™ is also coordinating the development of new public domain programs, and assembling texts and tutorials that describe the inner workings of the Amiga.

AMICUS™ is a place where Amiga users can distribute public domain software, in cooperation with other Amiga user groups. Also, AMICUS™ is a forum for discussion and documentation apart from the electronic networks. AMICUS™ has already assembled several disks of public domain programs in C, ABasic, and Amiga Basic.

These disks will be available through:

Amicus PDS
Amazing Computing
PiM Publications, Inc.
P.O. Box 869
Fall River, MA 02722

Due to the large demand for disks, it became necessary to coordinate the effort through this publication. The disks are available to everyone for \$7.00 (Amazing Computing subscribers may purchase the disks for \$6.00 each). Anyone purchasing disks is encouraged to distribute copies of the disks through their user groups and to their friends. This is Public Domain software and it is meant to be in as many hands as

possible.

AMICUS Beginnings

The AMICUS Network™ started in late summer on Usenet™, the worldwide Unix user's network. Originally for Amiga developers, AMICUS™ grew to include interested Amiga users, non-official software developers, and local Amiga user groups.

I have long had a dream that a national user group could produce great public domain software. There are few reasons why public domain software can't be as good as commercial software. So much great effort is put into public domain software, however, much of the effort is duplicated by different groups and individuals.

By coordinating ourselves, we can produce public domain spreadsheets, word processors, even compilers, and more. Creating new tools is an exciting way to learn about the Amiga™. All in all, AMICUS™ can speed and foster Amiga program development, both public and private.

There is a large base of existing C program source in the public domain. These programs were written for other computer systems, but can be modified and improved for use on the Amiga. A group like AMICUS™ can ease the conversion of these programs to the Amiga™.

By focusing our efforts, we can spend less time getting our Amigas up to speed. Although deciphering manuals is fun for some people, we can learn faster from each other. There are a lot of good programs to port, especially from the IBM PC™ and Unix™ environments.

If anyone has good program source, in any language, send it in! Remember, someone else might do the conversion. Many conversions only involve a few hours at the hands of a skilled programmer.

The AMICUS™ disks are archives of these converted programs. A major part of AMICUS™ will be a collection of texts and tutorials on aspects of the Amiga. So far, the public domain disks include tutorials on the CLI and Workbench, and C animation programming.

After all, people will be learning about the Amiga for many years to come, and these texts will be a valuable resource.

MicroEmacs, Arrange

The AMICUS™ disks include many useful programs. For example, a superb screen editor called MicroEmacs is in the public domain. Originally given away with Mark Williams C on the IBM-PC™, it has been ported to the Amiga, and will continue to improve as more and more features are added.

Also, a text formatting program called 'arrange' has been converted to the Amiga. While some people prefer a word processor that displays exactly what will be printed, others would rather edit their text with a simple text editor like ED.

A text formatting program like 'arrange' formats text files in this simple format according to imbedded commands. These commands set the margins, page

length, etc., and produce a formatted document. With a standalone program like 'arrange', you can edit your text in one window, and print the document in a background process.

IFF

Smart as they are, Commodore has announced that 'IFF', a format for storing graphic images, will be the recommended standard for exchanging sound and graphics data. It was developed at Electronic Arts, and will be used in all their programs. The IFF format will be available to the public, in order to promote this method of transferring data from program to program.

In other words, well-behaved Amiga graphics programs will have the same, interchangeable data format. Pictures designed with Deluxe Paint will be compatible with other programs, such as word processors. IFF will also encompass sounds created on the Amiga.

The IFF specification will be published in the V1.1 developer's manuals. Descriptions of IFF, with C code examples, can be found on AMICUS Public Domain Volume 2.

Amiga Networks

AMICUS™ members belong to networks like Usenet™, BIX™, CompuServe™, the Source™, the Well™, and the Fido™ net. I encourage members to spread the word about our software exchange - after all, we all benefit! There are several Amiga BBSs around the country. Their phone numbers are listed at the end of this column.

CompuServe has a new Amiga SIG. Type GO PCS-61. This group was generating several hundred messages a day at the outset. (ED note: see Bela Lubkin's article

elsewhere in this issue for access directions.)

PCS-155, under the sponsorship of the Toronto Pet User Group, has a sub-topic called Music and Graphics that has been dominated by Amiga discussion.

If you don't have a terminal program for your Amiga, the AMICUS Public Domain Volume 1 has one that can transmit at speeds greater than 1200 baud, and the C source code is included! This program, called ATerm, can also transfer files using the Xmodem protocol.

Quantum Link

In late 1985, Commodore announced Quantum Link, an information network, for users of the Commodore 64 and 128. Owners of these machines have been using Quantum Link for several months, and they report that the network is colorful and easy to use. Amiga owners will be able to use Quantum Link in the first quarter of 1986.

You will need Quantum Link's software to access this network, and the Amiga version won't be ready until early spring. This software allows transmission of graphics and color, file transfers, and supports such features as auto-dial and auto-login.

Quantum Link features a threaded message board system, much like the SIGs on CompuServe™. Users will also be able to upload and download software.

Price is the key feature of Quantum Link. The monthly fee of \$9.95 allows unlimited access, and downloading of software only costs 6 cents a minute. Quantum Link is only available after 6 p.m., though. But compare this price to other national networks - they charge around 20 cents a minute, non-prime-time.

If you would like more information about Quantum Link, call (703) 448-8700, or call 1-800-833-9400 with your 300 or 1200 baud modem.

Commodore also announced Amiga Link, a version of Quantum link for official Amiga developers. Amiga Link is similar in content to Quantum Link, but developers can contact the network with any terminal software. Commodore technical representatives will be available on-line for questions.

Amiga Bulletin Boards

On CompuServe, PCS-61 is the AMIGAForum. PCS-155 Subtopic 7 is pretty good, and PCS-44 has some Amiga users. Also try CBM2000, Commodore's own Special Interest Group.

First Amiga User Group

San Carlos, California (415) 595-5452
6 p.m. to 7 a.m. daily, 24 hours on weekends.

Welchnet

San Francisco, California (415) 644-2811

Amiga BBS

Denver, Colorado (303) 752-0247
300, 1200, 2400 baud

Casa Mi Amiga

Jacksonville, Florida (904) 733-4515

Amiga Advantage

New York (516) 661-4881

Super 68

New York, New York (212) 927-6919
300, 1200 baud

New York AMUSE

New York, New York (212) 269-4879

Maximillian Software BBS

California (408) 372-1722
300, 1200 baud, after 7 p.m., West Coast Time.

The Unknown TBBS

Denver, Colorado (303) 988-8155

•AC•

Public Domain Software

by
John Foust

If you are new to computing, perhaps you have not heard about public domain software. Yes, there are good programs for free or next to nothing. Perhaps you have seen glowing software guides that promise thousands and thousands of free programs. Well, it is not that easy, and public domain software is even harder to find for a new machine.

Often, the best programs take several months to spread across the country, and even then, it is hard to find someone with a copy of the program. Most public domain programs are circulated by user groups. But not everyone lives near a large city.

Here is where the AMICUS Network™ can help you. AMICUS members around the country will keep a watchful eye for new programs. They will send the latest and greatest Amiga public domain programs to Amazing Computing. After the program is tested and checked, it appears in the growing AMICUS public domain library.

Of course, the AMICUS Network needs more public domain programs. Everyone is encouraged to contribute programs.

Meanwhile, AMICUS™ will glean new software from every Amiga-oriented BBS in the country. The current public domain disks contain many ABasic programs. Some are C programs, and executable versions are provided for people who do not have C, but want to use the program.

If you belong to an Amiga user group, or know several Amiga owners, your group can get one copy of the AMICUS public domain disks, and then copy the programs yourself. If your local Amiga group has a public domain library, send a copy to AMICUS. For every new disk received, Amazing Computing will return a different disk from the Public Domain library.

Please, send the source code to your program. Remember, if you contribute the source code to your programs, others might improve it later. Also, source code is necessary to verify that the program does not contain serious bugs or malicious side effects. If you do not want to distribute the source code to the public, AMICUS still needs to see the source code to verify the safety of the program.

There are many public domain programs. Most of these programs were developed for other computers, from microcomputers to Unix mainframes.

KOTS

The AMICUS Network can serve to coordinate public domain projects. A good method to coordinate this kind of programming effort is called the "**Keeper of the Source**," or KOTS.

KOTS prevents the source code contention problems that plague all software development. For example, if three people independently improve a program called Paint!, and circulate their changes, as programs Paint! One, Paint! Two, and Paint! Three, some one else will

decide to integrate all three improvements, to make an even better program named Paint! Paint!. Now there are five versions of the Paint! program. Chances are, their files are now incompatible, so people who have the original Paint! program can't view pictures made with Paint! One, Paint! Two, and Paint! Paint!. This kind of problem has hampered public domain software from the beginning of time.

In the KOTS system, the source code for any program, such as the above terminal program, is maintained by a single person. This **"Keeper of the Source"** insures that there is only one official version of the code, even though the source code might be open to the public.

If someone has improved the code, they deal with the KOTS, and the KOTS coordinates the update. When the improved program is returned and checked, it supercedes the old code on the AMICUS public domain disks.

Obviously, it is better to confer with the KOTS before you start re-coding a program, to prevent wasted effort on all sides. People can customize a program for their own use, and never bother the KOTS. Someone could become a new KOTS if their program is substantially different from the original.

The AMICUS Network™ will maintain an informal list of KOTs. Not every program will need a KOTS. If this system is recognized among Amiga programmers, public domain software will evolve faster than before.

Terminal programs

A simple communications program is not very difficult, but a good one takes much more programming and forethought. People soon imagine extra features,

such as redial on busy, file transfers, automatic logon, and definable function keys.

A terminal program called ATerm, by Michael Mounier, is on AMICUS Volume 1. Aterm works at speeds over 1200 baud, and includes Xmodem file transfers.

Several AMICUS™ members are working on Kermit and CompuServe B file transfer programs. Bill Bond, the author of Macintosh FreeTerm, is now developing an Amiga terminal program, and plans to place it in the public domain.

Does anyone have source code for the VIDTEX or NAPLPS protocols? These videotext protocols allow the immediate transmission of graphic images.

A 'make' program

The 'make' program can be used in many ways, but the most popular use is to re-compile and link programs composed of many modules. Used in this way, 'Make' is similar to a smart EXECUTE file that only re-compiles modules that have changed since the last compile.

'Make' can be used to execute any series of CLI commands, but in a smarter fashion than 'EXECUTE'. For example, 'make' will be used to compile future AMICUS™ multi-part C programs, so even the newest C programmer can customize, compile and link a program.

Several versions of 'make' are available on the AMICUS disks, varying in extra features. A similar public domain program called 'cc.c' specializes in compiling C programs.

CLI command shell

A command shell would make the CLI easier to use. Are you tired of typing the same commands, again and again? For example, a command shell could allow you to edit your command line, using the right and left arrow keys to move about and correct mistakes. It could recall previous commands with the up and down arrow keys, to prevent re-typing.

A command shell can also perform other tasks, such as wildcard filename expansion. The command shell can keep track of its own variables, and make substitutions in the lines you type.

Text editors

The MicroEmacs editor is more full-featured than ED. George Jones, a SYSOP on CompuServe's AMIGAForum, provided the first port of MicroEmacs to the Amiga. It needs to be fine tuned to fit in with the Amiga's style. He said he will coordinate future updates.

At least one other editor, called Jove, is in the public domain. Jove is similar to Emacs, and current versions exist for the IBM PC. I have Jove source in Lattice C for PC-DOS, if any one is interested.

XLISP

This public domain Lisp has been converted to many other computers, and it will only be a matter of time before an Amiga version appears. Some people have discussed porting other languages to the Amiga, including the Icon language. I spoke with the U.S. distributor of COMAL here in Madison, and he said someone in Sweden is converting this language to the Amiga.

CLI tools

The standard CLI commands are inconsistent and hard to use. As someone said, "they are covered with hundreds of dinosaur footprints." Since all the CLI commands reside on disk, there is no reason we can't replace the CLI programs in SYS:c directory. Some CLI users prefer CLI to look like their favorite operating system - PC-DOS, Unix, TOPS-20, or VMS.

New CLI tools will undoubtedly be biased towards Unix-like tools, since C programmers tend to prefer this set of tools to all others, and first few new CLI tools will be written in C. Of course, these tools could be written in assembly language for speed and size, but those will take longer to evolve.

For starters, DECUS 'grep' is on the public domain disk, soon to be followed by other DECUS programs. 'Grep' searches text files for a given string of characters. DECUS is the DEC User Group, which has produced many public domain programs that will soon migrate to the Amiga.

The Workbench presents an interesting angle to this. Some people prefer to use the Workbench for everything. How can traditional CLI-type tools be integrated into the windowed style of program input and output?

How about a Workbench 'more' command, with nice gadgets to choose files, and view and scroll text? 'More' is a Unix program to display text files, similar to 'type /p' under PC-DOS. The Workbench really needs tools like this.

Spreadsheets

Turbo Pascal™ for PC-DOS included the public domain

source to a simple spreadsheet, and it is not hard to convert it to C. I started the conversion, but it remains unfinished. It looks easy to expand, too. I would rather work on it in C, even if Turbo Pascal has all the operating system hooks in the world. Does anyone have another spreadsheet source?

FORTH

I've heard of several forthcoming commercial versions of FORTH for the Amiga. Does anyone have the public domain 68000 source in assembler? Several versions exist for CP/M 68K.

Drawing programs

AMICUS™ Volume 1 has a copy of FreeDraw.c, a simple four-color Workbench drawing program in C. The author plans to expand it soon. Early CompuServe postings show that a good paint program is not hard to do in ABasic™, either, and several can be found on the same disk.

One AMICUS™ member hopes to write a BOB editor for animating larger objects. John Draper, also known as Captain Crunch, is developing a sprite editor for the Amiga. Draper also wrote a public domain tutorial on C animation programming, also on the AMICUS disks.

Since the Amiga has the standard IFF format for interchanging graphics data, it is possible for public domain programs to interact with commercial programs in a way that has not been seen before.

Word processors

TextCraft is an entry-level word processor. It is missing options that other word processors take for

granted, and TextCraft eats memory for lunch.

A text formatting program like 'nroff' might be better, and some versions are public domain. This type of word processing tool lets you use your favorite text editor to create ordinary ASCII files that include formatting commands.

A program of this type, called 'arrange', will be in the AMICUS library by the time you get this.

Printer definition files

While the standard Workbench has several common printer definitions, there is a substantial demand for less common printer definitions. A printer definition file allows the Amiga to print graphics and text to any printer.

The printer definition file can also make a dumb printer look smart, and make a smart printer look brilliant. The specification for Amiga printer definitions allows custom program code to be executed when a particular escape sequence is sent to PRT: device. In this way, a printer definition is really a program. This can really fine tune a printer definition, and let a less intelligent printer act smart.

The AmigaDOS version 1.1 manuals will tell more about printer definition files. One AMICUS developer reports troubles making new printer definitions, but that should change when the updates manuals are released, since he was using a pre-release copy of the version 1.1 manual.

MIDI interfaces

MIDI interfaces for the Amiga are available for under

\$50 from Cherry Lane Technologies. They connect to the serial port. At one time, the Preferences tool actually had a MIDI setting in the Baud Rate box. You will also find MIDI listed in the index of your Amiga manual, but that page doesn't say anything about MIDI.

Don't be fooled by Atari ST™ owners who think the ST is superior to the Amiga in this respect. Although the MIDI port is present on the standard ST™, a good MIDI program is hard to write, and I am sure both machines are equally capable in this regard. The Amiga serial port can handle the high speed of MIDI data with ease.

Sounds

Good sounds are hard to generate in ABasic. One alternative I have considered works like this: Tables of actual, digital sampled sounds are saved to disk, ready to be loaded and used by any program. In ABasic, this data could be loaded directly into the WAVE arrays. These sounds could be collected on a public domain disk.

If you are interested in the format of sound tables, see the text called 'soundform' on AMICUS Volume 1.

IBM emulator programs

The PC emulator opens up another world of public domain software. AMICUS™ will also have disks of popular public domain programs that work with the PC emulator, in the 5 1/4 and 3 1/2 disk formats. If you have a favorite PC public domain program, send AMICUS a note, and we will transfer it to these formats.

AMICUS and you

What would you like AMICUS to be? I see it as a worldwide Amiga network, a place where Amiga users can gather and distribute public domain software, in

cooperation with other Amiga users and user groups. I hope to see members contribute articles to this magazine, and this column, and start a forum for discussion and documentation apart from the electronic nets.

Hopefully, there will be a cross-fertilization of ideas between programmers of the Macintosh™, the Atari ST™, and the Amiga™. AMICUS™ can help foster this contact between different user groups, and reduce the unnecessary bickering and competition between owners of these computers. It is almost as if every computer owner has to become a public relations agent to fend off criticism.

I am certain you will think of more programs. Many conversions only involve a few hours work at the hands of a skilled programmer. If you have any good program source, in any language, send it in. Remember, someone else might want to do the conversion.

If you have the time and skill to convert programs, or have ideas you would like to share with the group, please write a letter to me in care of Amazing Computing™.

It is very possible for a novice to dream up a new type of computer program, but not everyone has the ability to craft an idea into code. Computer gurus do not have a corner on inventiveness. I will summarize everyone's ideas in this column in the months to come.

To obtain Public Domain disks from the AMICUS library, send a disk of your own collected software, if it is new to Amicus, we will return a disk from the library.

If you do not have a disk of your own, send \$7.00 for

each disk ordered (\$6.00 for Amazing Computing subscribers) to:

AMICUS PDS
PiM PUBLICATIONS, Inc.
P.O. Box 869
Fall River, MA. 02722

Currently there are two disks in the library:

- 1 AMICUS developer disk containing C source code.
- 2 Abasic disk with programs and utilities.

There will be more disks collected and their contents

listed in this section each issue. Come and join, be a part of a new frontier.

John Foust is an Amiga developer for Sight & Sound Music Software in Madison, Wisconsin. His CompuServe ID is 72237,135, his Source ID is BDN338. John Foust may be reached through the above sources or by mail to:

John Foust/AMICUS™
PiM Publications
P.O. Box 869
Fall River, MA 02722

•AC•

COMING NEXT MONTH!

In

Amazing Computing

A continuation of the three tutorials:

The Amazing C Tutorial
Lisp Tutorial
Inside CLI (with an investigation of ED,EDIT, and EXEC)

Deluxe Paint™ as well as the new Amiga™ games from Electronic Arts.

An examination of AmigaBasic™

Research of the new terminal programs for the Amiga™ including OnLine™.

ROOMERS

Miga-Maniac.

New software and hardware releases for the Amiga™.

New Amiga™ public domain software

Plus, John Foust will return with the **AMICUS™ Network**

In short, we are preparing all of the above, plus several surprises for another great issue. See you then.



Marauder

Backup your valuable software
with Marauder!

Marauder lets you read, write, analyze
and edit raw disk data.

Marauder picks up where DiskCopy
and DiskEd leave off.

The Amiga tool you can't afford
to do without!

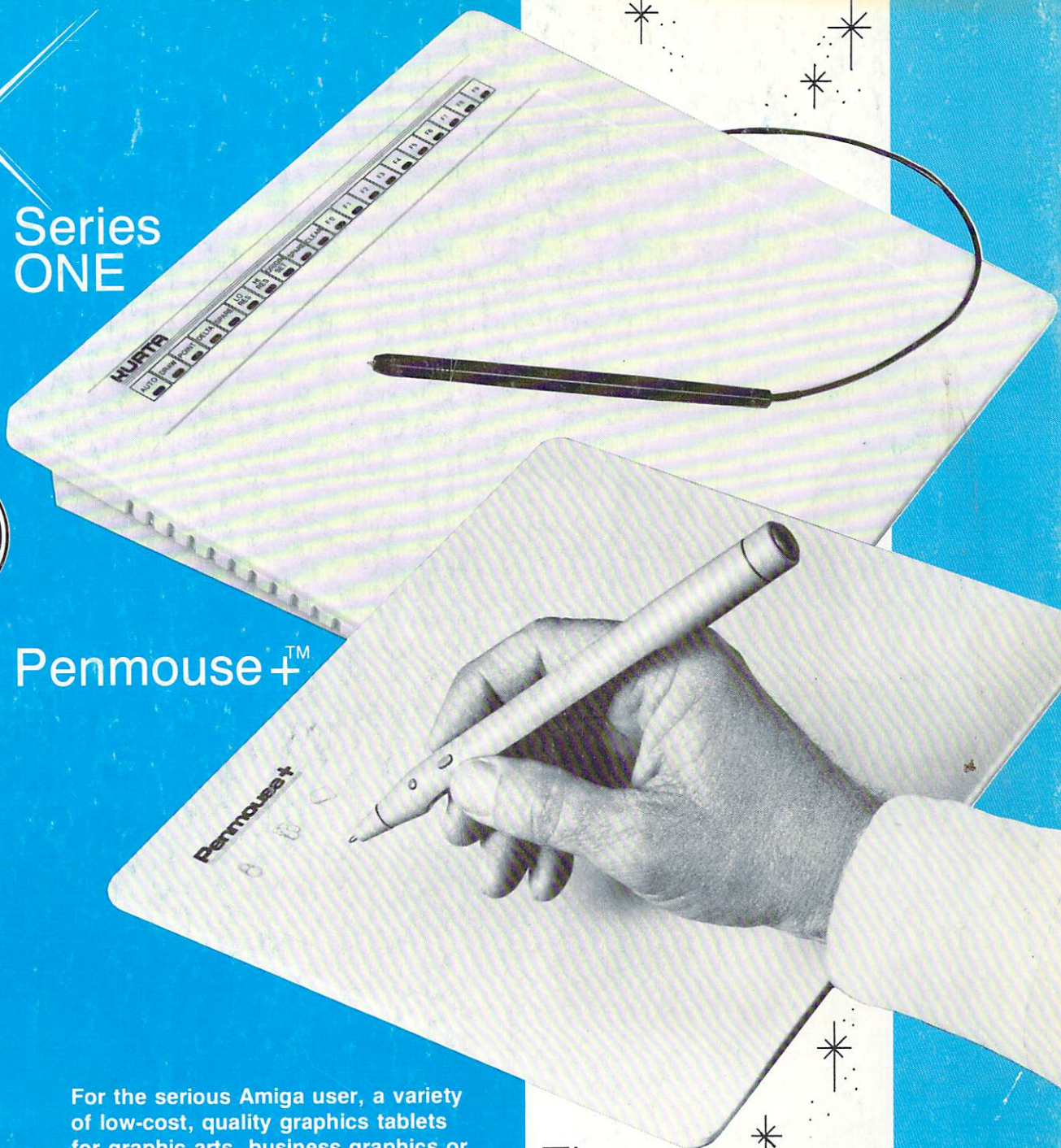
Available soon from:

Discovery Software
262 S. 15th Street - Suite 300
Philadelphia, PA 19102
Phone: (215) 546-1533

Dealer Inquiries Welcome

Future
with
Kurta

Series
ONE



Penmouse+™

For the serious Amiga user, a variety of low-cost, quality graphics tablets for graphic arts, business graphics or CAD/CAM applications — from Kurta.

The ergonomically sloped Series ONE tablet, with resolution of up to 200 PPI and a built-in power supply, is available in three sizes: 8.5" x 11", 12" x 12", and 12" x 17"

The Penmouse+™ graphics tablet input device is an innovative new product with the features of both a tablet and a mouse. This versatile system comprises a cordless, battery-powered pen and a 1/4" thin tablet — both at an extremely low cost.

Get the most out of your Amiga! See the Kurta graphics tablets at your nearest Amiga dealer or contact . . .

The
Leaders in
Innovative
Graphic
Systems

KURTA®
CORPORATION

4610 South 35th Street
Phoenix, AZ 85040
(602) 276-5533